Automatic Phoneme Recognition of South African English

Herman A. Engelbrecht

Thesis presented in partial fulfilment of the requirements for the degree
Master of Science in Electronic Engineering
at the University of Stellenbosch

Supervisor: Prof J.A. du Preez

April 2004

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work, except where stated otherwise.

SIGNATURE DATE

Abstract

Automatic speech recognition applications have been developed for many languages in other countries but not much research has been conducted on developing Human Language Technology (HLT) for S.A. languages. Research has been performed on informally gathered speech data but until now a speech corpus that could be used to develop HLT for S.A. languages did not exist. With the development of the African Speech Technology Speech Corpora, it has now become possible to develop commercial applications of HLT.

The two main objectives of this work are the accurate modelling of phonemes, suitable for the purposes of LVCSR, and the evaluation of the untried S.A. English speech corpus.

Three different aspects of phoneme modelling was investigated by performing isolated phoneme recognition on the NTIMIT speech corpus. The three aspects were signal processing, statistical modelling of HMM state distributions and context-dependent phoneme modelling. Research has shown that the use of phonetic context when modelling phonemes forms an integral part of most modern LVCSR systems. To facilitate the context-dependent phoneme modelling, a method of constructing robust and accurate models using decision tree-based state clustering techniques is described. The strength of this method is the ability to construct accurate models of contexts that did not occur in the training data. The method incorporates linguistic knowledge about the phonetic context, in conjunction with the training data, to decide which phoneme contexts are similar and should share model parameters.

As LVCSR typically consists of continuous recognition of spoken words, the context-dependent and context-independent phoneme models that were created for the isolated recognition experiments are evaluated by performing continuous phoneme recognition. The phoneme recognition experiments are performed, without the aid of a grammar or language model, on the S.A. English corpus. As the S.A. English corpus is newly created, no previous research exist to which the continuous recognition results can be compared to. Therefore, it was necessary to create comparable baseline results, by performing continuous phoneme recognition on the NTIMIT corpus. It was found that acceptable recognition accuracy was obtained on both the NTIMIT and S.A. English corpora. Furthermore, the results on S.A. English was $2-6\%$ better than the results on NTIMIT, indicating that the S.A. English corpus is of a high enough quality that it can be used for the development of HLT.

Synopsis

Automatiese spraak-herkenning is al ontwikkel vir ander tale in ander lande maar, daar nog nie baie navorsing gedoen om menslike taal-tegnologie (HLT) te ontwikkel vir Suid-Afrikaanse tale. Daar is al navorsing gedoen op spraak wat informeel versamel is, maar tot nou toe was daar nie 'n spraak databasis wat vir die ontwikkeling van HLT vir S.A. tale. Met die ontwikkeling van die African Speech Technology Speech Corpora, het dit moontlik geword om HLT te ontwikkel vir wat geskik is vir kommersiële doeleindes.

Die twee hoofdoele van hierdie tesis is die akkurate modellering van foneme, geskik vir groot-woordeskat kontinue spraak-herkenning (LVCSR), asook die evaluasie van die S.A. Engels spraak-databasis.

Drie aspekte van foneem-modellering word ondersoek deur isoleerde foneem-herkenning te doen op die NTIMIT spraak-databasis. Die drie aspekte wat ondersoek word is sein prosessering, statistiese modellering van die HMM toestands distribusies, en konteks-afhanklike foneem-modellering. Navorsing het getoon dat die gebruik van fonetiese konteks 'n integrale deel vorm van meeste moderne LVCSR stelsels. Dit is dus nodig om robuuste en akkurate konteks-afhanklike modelle te kan bou. Hiervoor word 'n besluitnemings-boom-gebaseerde trosvormings tegniek beskryf. Die tegniek is ook in staat is om akkurate modelle te bou van kontekste van nie voorgekom het in die afrigdata nie. Om te besluit watter fonetiese kontekste is soortgelyk en dus model parameters moet deel, maak die tegniek gebruik van die afrigdata en inkorporeer taalkundige kennis oor die fonetiese kontekste.

Omdat LVCSR tipies is oor die kontinue herkenning van woorde, word die konteks-afhanklike en konteks-onafhanklike modelle, wat gebou is vir die isoleerde foneem-herkenningseksperimente, evalueer d.m.v. kontinue foneem-herkening. Die kontinue foneem-herkenningseksperimente word gedoen op die S.A. Engels databasis, sonder die hulp van 'n taalmodel of grammatika. Omdat die S.A. Engels databasis nuut is, is daar nog geen ander navorsing waarteen die resultate vergelyk kan word nie. Dit is dus nodig om kontinue foneem-herkennings resultate op die NTIMIT databasis te genereer, waarteen die S.A. Engels resulte vergelyk kan word. Die resulate dui op aanvaarbare foneem herkenning op beide die NTIMIT en S.A. Engels databassise. Die resultate op S.A. Engels is selfs $2 - 6\%$ beter as die resultate op NTIMIT, wat daarop dui dat die S.A. Engels spraak-databasis geskik is vir die ontwikkeling van HLT.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

ASR	Automatic speech recognition
AST	African Speech Technology
CA	Cepstral analysis
CART	Classification and regression trees
CMS	Cepstral mean subtraction
DC	Diagonal covariance
DTFT	Discrete-time fourier transform
DTW	Dynamic time warping
EM	Expectation maximisation
FC	Full covariance
GMM	Gaussian mixture model
HLT	Human language technology
HMM	Hidden Markov model
HTK	Hidden Markov model ToolKit
IDTFT	Inverse discrete-time fourier transform
IID	Independently and identically distributed
ISL	Interactive Systems Laboratory
IVR	Interactive voice response
LC	Left-context
LDA	Linear discriminant analysis
LP	Linear prediction
LPA	Linear predictive analysis
LPCC	Linear predictive coding epstral coefficients
LVCSR	Large-vocabulary continuous speech recognition
MFCC	Mel-frequency cepstral coefficients
MLE	Maximum likelihood estimation
PER	Phoneme error rate
PLP	Perceptual linear predictive cepstral coefficients

SAMPA	Speech Assessment Methods Phonetic Alphabet
RC	Right-context
VQ	Vector quantisation

Symbols

$\alpha_t(i)$	Forward probability
$\beta_t(i)$	Backward probability
$\gamma_t(i)$	Probability of occupying HMM state i at time t given the observation sequence $\mathbf{X}$
Δs	Velocity feature vectors
$\Delta\Delta$s	Acceleration feature vectors
$\Delta\mathcal{L}(s,q)$	Log-likelihood splitting criteria of splitting leaf-node Ψ_s^k with question $Q(q)$
η	Minimum increase in log-likelihood threshold
μ	Mean vector
ν	Preemphasis filter coefficient
π_i	Initial probability of HMM state i
Φ	Hidden Markov model
Φ_ρ	Hidden Markov model of phoneme
Φ_u	Hidden Markov model of utterance
Ψ	Decision tree node
ρ	Context-independent base phone
$\mathbf{\Sigma}$	Covariance matrix
τ	Minimum node occupancy threshold
$\theta(n)$	Simplified discrete-time vocal tract impulse response
Θ_0	Gain of discrete-time speech production model
$\Theta(z)$	Discrete-time speech production model
$\hat{\Theta}_0$	Gain of estimate of discrete-time speech production model
$\hat{\Theta}(z)$	Estimate of discrete-time speech production model
ω	Frequency [rad $\cdot$ s^{-1}]
$\zeta_t(i,j)$	Probability of making the transition from state i to state j at time $t+1$, given the observation sequence $\mathbf{X}$
$\mathbf{\Omega}$	Eigenvalue matrix
$\mathbf{A}$	State probability transition matrix of a HMM
a_{ij}	HMM state i to state j transition probability
a_i	i$^{\text{th}}$ numerator coefficient of the speech production model $\Theta(z)$
$\hat{a}_i$	Estimate of LP filter parameters

$\mathbf{B}$	State output probability distributions of a HMM
b_i	i^{th} denominator coefficient of speech production model $\Theta(z)$
$\mathcal{B}$	Intermediate rotation matrix of LDA
c_{jk}	k^{th} mixture weight of output GMM distribution of state j
$c_s(n)$	Real cepstrum of discrete-time speech signal $s(n)$
$\mathcal{C}_\rho$	Allophone derived from context-independent base phone ρ
$\mathbb{C}_\rho$	The set of unique allophone contexts $\mathcal{C}_\rho$ derived from the base phone ρ
$\mathbb{C}_\rho(s)$	The subset of unique allophone contexts $\mathcal{C}_\rho$ associated with leaf-node Ψ_s^k
D	Dimension of acoustic data $\mathbf{X}$
$\mathcal{D}_i(s,q)$	Index to the i^{th} descendent formed by splitting leaf node Ψ_s^k with splitting question $\mathbb{Q}(q)$.
$\mathbb{D}^*$	Descendents of splitting optimal leaf node $\Psi_{s^*}^k$
$e(n)$	Discrete-time LP error signal
$E^{\mathcal{C}_\rho}$	Number of training examples of allophone $\mathcal{C}_\rho$
$\mathcal{F}(\cdot)$	Discrete-time Fourier transform (DTFT)
$\mathcal{F}^{-1}(\cdot)$	Inverse discrete-time Fourier transform (IDTFT)
$G(z)$	Glottal pulse model
h_k	k^{th} filter coefficient of vocal tract model $H(z)$
$H(z)$	Vocal tract model
H_0	Gain of vocal tract model $H(z)$
K	Vocal tract filter order
$\mathbf{K}$	Dimension-reduction transformation matrix of LDA
k	Index to HMM states
L	Length of discrete-time speech signal $s(n)$
$\mathbb{L}$	Set of leaf-nodes of a decision tree
$\mathcal{L}(\cdot)$	General log-likehood
M	LP filter order
$\mathcal{M}$	Number of mixtures in a GMM or tied-mixture distribution
N	Number of states in a hidden Markov model (HMM)
$N_{\mathbb{L}}^k$	Number of leaf-nodes in decision tree $\mathcal{T}_\rho^k$, of k^{th} state of base phone ρ
N_ρ	Number of allophone contexts of base monophone ρ
N_F	Number of HMM terminating states
$N_{\mathbb{Q}}$	Number of decision tree questions
$\mathcal{N}(\cdot)$	Normal (or Gaussian) probability distribution
$\mathcal{N}_{\mathrm{shared}}(\cdot)$	Shared normal (or Gaussian) probability distribution
$\mathbb{N}$	The set of natural numbers
$\mathbb{N}_{\mathrm{shared}}$	Set of shared normal (or Gaussian) probability distributions
$\mathbb{N}_{\mathrm{unshared}}$	Set of unshared normal (or Gaussian) probability distributions

O	Number of zeroes of discrete-time speech production model $\Theta(z)$
p_i	*A priori* probability of class i
P	Pitch period of excitation sequence (impulse generator)
P_s	Power of discrete-time speech signal $s(n)$
$P_r(z)$	Preemphasis filter
$P(\cdot)$	General probability
$\mathcal{P}$	Parent node in a decision tree
s^*	Index to optimal leaf node in the decision tree to split
$Q(q)$	A single decision tree question
$Q(q^*)$	The best question to split leaf node Ψ_{s^*}
q^*	Index to the best question to split Ψ_{s^*}
q_j	j^{th} hidden Markov-model state
$\mathbf{Q}$	Hidden Markov-model states
$\mathbf{Q}(\Phi, \hat{\Phi})$	Baum's auxiliary function
$\mathbb{Q}$	Set of decision tree questions regarding linguistic context
R	Number of poles of discrete-time speech production model $\Theta(z)$
$R(z)$	Radiation model
s_t	Hidden Markov-model state occupied at time index t
$\mathbf{S}$	Hidden state sequence
$\mathcal{S}$	An unspecified splitting criteria
$\mathbb{S}_\rho^k$	Set of k^{th} tied HMM states of the allophones in the cluster $\mathbb{C}_\rho(s)$
$S(\omega)$	Fourier transform of speech signal $s(n)$
$s(n)$	Discrete-time speech signal
$\hat{s}(n)$	Estimate of discrete-time speech signal $s(n)$
$s_{\text{pnorm}}(n)$	Power normalised speech signal
$tr(\cdot)$	Trace
t	Time index
T	Length of the observation sequence $\mathbf{X}$
$\mathcal{T}_\rho^k$	A decision tree grown to share the k^{th} HMM states of base phone ρ
$\mathcal{T}$	Time-aligned labels of the acoustic data $\mathbf{X}$
$u(n)$	Discrete-time excitation signal
$u_{\text{glottis}}(n)$	Discrete-time glottal excitation signal
$\mathbf{U}$	Eigenvector matrix
$V_t(i)$	Best-path probability
$\mathcal{V}$	Vocabulary of phonemes
$\mathbf{W}$	Spoken phoneme or word sequence
$\mathcal{W}$	Phoneme or word alphabet
w_i	i^{th} word in phoneme or word sequence

$\widehat{\mathbf{W}}$	Most likely word or phoneme sequence
$\mathbf{X}$	Acoustic speech features or observation sequence
$\mathcal{X}$	Acoustic speech features alphabet
$\mathbf{x}_i$	i^{th} vector in acoustic data sequence
$\mathbf{Y}$	Feature vectors whose dimensions are uncorrelated
z_0	Radiation model $R(z)$ filter coefficient

Chapter 1

Introduction

1.1 Motivation

The fact that South Africa is still a developing country implies that population sectors exist that have never had any access to computers or computer based technology. The financial cost of this type of technology could be blamed. Even if access is made available, mastering the required skills to utilise this technology via the conventional way, is not easy. For all population sectors to benefit from computer based technology it is necessary to allow easy access without having to learn difficult skills such as typing. One could therefore argue that, easy access to Human Language Technology (HLT), such as Automatic Speech Recognition (ASR), could be the bridge across this digital divide.

At this stage HLT has not matured sufficiently to allow easy access to technology. Although ASR applications have been developed for many languages in other parts of the world, not much research has been conducted on developing HLT for South African languages. Research has been performed on informally gathered speech data, such as the research conducted on isolated word recognition of isiXhosa [8], but until now a speech corpus that could be used to develop HLT for South African languages did not exist. With the development of the African Speech Technology Speech Corpora (Appendix C), it has now become possible to develop commercial applications of HLT.

The purpose of this research was to develop the acoustic models needed for the automatic speech recognition component of a prototype hotel booking system. The purpose of the prototype is to demonstrate that HLT for South African languages can be developed locally and that it is comparable to applications which has been developed abroad.

1.2 Background

This thesis is concerned with a specific branch of speech recognition known as *statistical speech recognition*. From this point on, references to speech recognition actually refers to

statistical speech recognition. A modern statistical speech recogniser has the following components:

- Acoustic Processor
- Acoustic Models
- Language Models
- Hypothesis Search

In order to better understand the breakup of a modern speech recogniser into its components, as well as to understand the separate components, it is useful to examine the mathematical formulation of speech recognition (as given by Frederick Jelinek [23]).

1.2.1 A mathematical formulation of speech recognition

Let $\mathbf{X}$ denote the acoustic speech features on which the recogniser will base its decision about which words were spoken. Because speech recognition is performed on digital computers, it can be assumed that $\mathbf{X}$ is a sequence of symbols from some alphabet $\mathcal{X}$:

$$\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m \quad \mathbf{x}_i \in \mathcal{X} \tag{1.1}$$

Let $\mathbf{W}$ denote the string of n words (or phonemes), each belonging to a fixed and known vocabulary $\mathcal{W}$:

$$\mathbf{W} = w_1, w_2, \ldots, w_n \quad w_i \in \mathcal{W} \tag{1.2}$$

$P(\mathbf{W}|\mathbf{X})$ is the probability that the words $\mathbf{W}$ were spoken, given that the acoustic speech features $\mathbf{X}$ were observed. Assuming that all the words are equally important[1] and that a maximum likelihood recogniser is being used, the recogniser should choose the most likely word string, $\hat{\mathbf{W}}$, given the observed acoustic data $\mathbf{X}$. Therefore

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{W}|\mathbf{X}) \tag{1.3}$$

Using Bayes' formula of probability theory the conditional probability of $\mathbf{W}$, given $\mathbf{X}$, is rewritten as

$$P(\mathbf{W}|\mathbf{X}) = \frac{P(\mathbf{W})P(\mathbf{X}|\mathbf{W})}{P(\mathbf{X})} \tag{1.4}$$

where $P(\mathbf{W})$ is the probability that the word string $\mathbf{W}$ is spoken, $P(\mathbf{X}|\mathbf{W})$ is the probability that when $\mathbf{W}$ is spoken the acoustic data $\mathbf{X}$ will be observed, and $P(\mathbf{X})$ is the

[1] The words being recognised are assumed to come from normal conversational speech and the purpose is not to correctly recognise only important words such as 'Help!'

average probability that $\mathbf{X}$ will be observed. Since the maximisation in Eq. 1.3 is carried out with fixed acoustic data $\mathbf{X}$, it follows from 1.3 and 1.4 that the recogniser must find the string $\hat{\mathbf{W}}$ that maximises the product $P(\mathbf{W})P(\mathbf{X}|\mathbf{W})$. Equation 1.3 can consequently be simplified to

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{W})P(\mathbf{X}|\mathbf{W}) \tag{1.5}$$

1.2.2 Components of a speech recogniser

The components of a modern speech recogniser can be readily distinguished by examining Eq. 1.5:

1.2.2.1 Acoustic processor

Speech is a continuous pressure waveform, but a spoken phrase can be represented as a sequence of discrete acoustic units (typicall phonemes). In order to recognise speech, the continuous speech waveform needs to be transformed into a sequence of discrete acoustic units. Because the goal is to perform *automatic* speech recognition, it can be safely assumed that some form of software performs this transformation. If a computer is being used, a microphone is used to tranduce the speech from a pressure waveform to an electrical signal. This electrical signal is then sampled (using some form of analog-to-digital conversion) to provide a one-dimensional sequence of samples $s(n)$. The purpose of the acoustic processor, or front-end, is to transform the sampled speech signal, $s(n)$, into the set of acoustic speech features $\mathbf{X}$. The specific acoustic processor used in this research is discussed in Chapter 2.

1.2.2.2 Acoustic models

The acoustic processor provides the rest of the recogniser with acoustic speech features $\mathbf{X}$. The rest of the recogniser needs to compute the $P(\mathbf{X}|\mathbf{W})$ — the probability that when the speaker produces the sentence $\mathbf{W}$, the acoustic processor produces the acoustic speech features $\mathbf{X}$. This thesis is concerned with the accurate modelling of phonemes that are required for speaker-independent, Large Vocabulary Continuous Speech Recognition (LVCSR). It is therefore safe to assume that for each spoken sentence $\mathbf{W}$, the acoustic speech features $\mathbf{X}$ will be different. Thus it becomes necessary to be able to compute $P(\mathbf{X}|\mathbf{W})$ *on-the-fly*. It also becomes necessary to construct acoustic models for each word (or phoneme) $w_i \in \mathcal{W}$ beforehand. At this stage the distinction between word and phoneme models is not made, but is it important to remember that this thesis is concerned with the automatic recognition of *phoneme* sequences and *not* word sequences. Hidden Markov Models (HMMs) have been used to model these acoustic models and in Chapter 3

the theoretical foundation of HMMs is laid. Chapter 4 will discuss the acoustic modelling of context with HMMs and also the problems associated with it.

1.2.2.3 Language modelling

Referring back to Eq. 1.5, the only term which remains is $P(\mathbf{W})$, the *a priori* probability that the speaker wishes to utter the string $\mathbf{W}$. Although it is not computationally feasible to estimate $P(\mathbf{W})$ for every possible word string $\mathbf{W}$, it has been shown that the use of simple language models greatly decreases the word error rate of speech recognisers. It is important to realise that the language model to be used for the recogniser depends on the application of the recogniser. For example, the set of word strings that a lawyer uses in a courtroom differs greatly from the set used by an engineer researching speech recognition.

According to [24] the branching factor of a language or grammar is an important measure of the degree of difficulty of a specific speech recognition task. The estimate of the branching factor is called the *perplexity*, since it describes how confusing the grammer (or language) is. It is generally true that lower perplexity correlates with better recognition performance. The higher the perplexity, the more branches the speech recogniser needs to consider statistically. Therefore a recogniser that achieves a high recognition accuracy on a recognition task with a low perplexity is not necessarily better than a recogniser that achieves a lower recogniton accuracy on a task with a higher perplexity.

The phoneme models produced by this research will be used in a prototype Interactive Voice Response (IVR) hotel booking system. As such, the anticipated grammar is very restricted and therefore the number of allowable word strings $\mathbf{W}$ is much smaller than the number of word strings in normal speech. Because the language modelling will be provided by another part of the hotel booking system, this research is focused on the acoustic modelling of the phonemes. For this reason no language model is used when performing the phoneme recognition experiments.

1.2.2.4 Hypothesis search

Lastly the recogniser needs to find the correct transcription $\hat{\mathbf{W}}$ of the acoustic data $\mathbf{X}$. One method would be to calculate $P(\mathbf{W})P(\mathbf{X}|\mathbf{W})$ for every possible word string $\mathbf{W}$ and to subsequently pick the word string with the highest probability. Unfortunately it is not feasible to search through all the possible word strings. It is therefore necessary to only examine those word strings somehow suggested by the acoustic data $\mathbf{X}$. This will be discussed in more detail in Chapter 3. Another method of making the hypothesis search feasible is to limit the recogniser to a very specific domain of speech (i.e. a lawyer in a courtroom). This has the effect of reducing the infinite number of possible word strings to a small set of allowable word strings and therefore we can compute the likelihood of

all the allowable word strings. The only problem with this method is that the recogniser could then *only* be used in the specific domain of speech for which it was designed.

1.3 Prior work on speech recognition

Extensive research on the topic of isolated- and continuous word recognition has been conducted over the years. At first whole-word models were constructed for each word in a limited vocabulary and used for isolated word recognition. Whole-word models achieved acceptable recognition accuracies, but it became evident that one would need a large number of word models when moving to LVCSR systems (the English language alone contains in the order of millions of different words). In order to construct the whole-word models it would be necessary to have a large number of occurrences of each word in the transcribed speech corpus and therefore one would need huge quantities of data. Obtaining these huge speech corpora is prohibitively expensive and the conclusion is that constructing whole-word models for LVCSR is not feasible. The solution to this problem is to model a relatively small number of sub-word speech units, which are then used as building blocks to construct word models. The amount of data needed to construct the sub-word models is orders of magnitude smaller than the amount of data needed to construct whole-word models. For this reason speech recognition research moved in the direction of sub-word modelling, and specifically phoneme modelling.

At first context-independent phoneme models were used as building blocks for word models. Depending on the application language there could be between 40 and 70 of these phoneme models. It was found that arbitrary word models could be constructed, but that the recognition accuracy of the sub-word based word models was not as good as that of the whole-word models. This is due to *co-articulation* (which is explained in more detail in Chapter 4) and requires the use of context-dependent phoneme modelling to counter its detrimental effects.

Context-dependent phoneme models was first introduced by IBM and BBN. A system based on their early techniques only resulted in a mediocre inhouse word recognition accuracy of 58% in speaker independent recognition [26]. A great breakthrough in phoneme recognition (and therefore continuous speech recognition) was made in 1989 by Kai-Fu Lee with his development of the SPHINX system. Using context-dependent phoneme models, the SPHINX system was able to achieve a phoneme correct rate of 73.8% and a word recognition accuracy of 94% on the Resource Management Task, for a perplexity of 60 and using a word-pair grammar. Kai-Fu Lee demonstrated that one could use context-dependent phoneme models in order to perform automatic speech recognition. By developing the SPHINX system Kai-Fu Lee opened the door to practical LVCSR systems.

The problem with the use of context-dependent models is chiefly a lack of enough

training data and the construction of contexts that did not occur in the training data, resulting in an insufficient number of observed contexts. In the SPHINX system the data scarcity problem was addressed in two ways: the first was to use deleted interpolation to smooth the specific, but poorly estimated, context-dependent phoneme models with general, well-estimated context-independent models. The second was to merge phoneme models with similar contexts. In 1991 Bahl *et al.* [1] proposed using decision trees to model the context-dependent phonemes thereby reducing the total number of models. In 1993 Yunxin Zhao also addressed the data scarcity problem by using model smoothing and achieved a continuous word recognition accuracy of 90.9% for a perplexity of 24 and using a word-pair grammar [53].

All the work done up until the end of 1992 addressed the data scarcity problem by smoothing the context-dependent models using less context-dependent models and context-independent models. Mei-Yuh Hwang and Xuedong Huang [19] then came up with the idea of the sharing (or tying) of the subphone states (or *senones* as they called these subphone states) of the hidden Markov model. These states thus form state clusters, with each cluster sharing common subphone states. They achieved a word error rate of only 3.8% (recognition accuracy of 96.2%) on the DARPA resource management task of February and October 1989. In 1994 Young, Woodland and Odell combined the ideas of decision trees and *senones* [51], [52], [31]. They achieved a phone recognition accuracy of 72.3% using a bigram language model on the TIMIT speech corpus and a word recognition accuracy of 95.9% and higher on the DARPA Resource Management tests of February 1989 to September 1992. In 1996 Mei-Yuh Hwang, Xuedong Huang and Fileno Alleva suggested a possible solution for the unseen triphone problem by using decision trees and *senones* [20].

Most of the research of the past five years have been concerned with improving the clustering (or merging) of distributions for the *senones*. Recently a paper was published which proposed a Bayesian approach to triphone construction [30], which is based on Bayesian statistics rather than heuristic methods such as state clustering. According to the research the Bayesian triphone results in higher accuracies than the triphones constructed from the heuristic methods.

Most of the current speech recognition research have been performed using a pronunciation lexicon and a language model. The pronunciation lexicon indicates the possible phoneme sequences that words consist of. The language model indicates the probability that certain words (or phonemes) follow one another. The benefit of pronunciation lexicons and language models are that they enable speech recognition systems to achieve high word recognition accuracies. The reason for this is that the pronunciation lexicon and the language model limit the possible sequences of phonemes.

Several research studies conducted during the past ten years have focused on improving

pronunciation lexicons and language models [23], [24]. However, this thesis is concerned with phoneme recognition without the aid of a lexicon or a language model. The literature on such research is sparse but recent references to phoneme recognition without the aid of a language model was reported by the Interactive Systems Laboratory (ISL) of Carnegie Mellon University [42, 43, 45, 46]. ISL used the GlobalPhone speech corpus ([41, 47]) which is high-quality read speech similar to the TIMIT speech corpus [48]. They reported a phoneme error rate of 46.4% with a vocabulary of 46 phonemes and a trigram perplexity of 9.2.

The question may be asked whether performing phoneme recognition experiments, without any language model or grammar, is a good method of evaluating phoneme-modelling techniques? A more useful evaluation would be to perform phoneme-based continuous-word recognition experiments, utilising all the techniques (such as language models, fast-matching searches, multi-pass recognition, etc.) that are necessary to obtain acceptable continuous speech recognition accuracy. Unfortunately, this requires a speech corpus suitable for large vocabulary continuous-word recognition experiments. Futhermore, performing continuous word recognition experiments requires a fair amount of extra techniques in order to obtain acceptable recognition, which makes it more difficult to determine the influence of the phoneme modelling techniques on the recognition accuracy. Lastly, the development of a complete large vocabulary continuous speech recognition system is not a trivial task.

1.4 Objectives

The primary objects for this project were identified as:

- The accurate modelling of phonemes that are suitable for the purposes of LVCSR. This includes the secondary objectives:
  - An evaluation of signal analysis techniques suitable for phoneme modelling in terms of accuracy, sensitivity towards data scarcity, robustness and computational efficiency.
  - The selection and application of appropriate statistical modelling techniques for the estimation of phoneme models, as required by LVCSR systems, with specific focus on contextual effects.
- The evaluation of the previously untried South African English speech corpus. This includes the secondary objectives:
  - The determination of the suitability of the speech corpus to the development of HLT applications. The suitability depends on the performance of phoneme

models created from the speech corpus.

- The verification that the phoneme modelling techniques (which result in acceptable recognition performance on American and British English) result in acceptable recognition performance of South African English over the telephone network.

1.5 Contributions

- This work is the first application of phoneme modelling to professionally gathered South African English and has shown that the South African English speech database is of an acceptable quality.
- This research has shown that acceptable recognition performance of South African English can be achieved over the South African telephone network.
- This research has shown that extracting Perceptual Linear Predictive Cepstral Coefficients (PLPs) is an acceptable feature-extraction alternative to the more well-known extraction of Mel Frequency Cepstral Coefficients (MFCCs).
- It is generally accepted that, in order to obtain faster recognition speeds, it is better to use diagonal-covariance rather than full-covariance Gaussian Mixture Models (GMMs) as HMM state distributions. By comparing diagonal-covariance and full-covariance GMM state distributions on an equal-parameter basis, it has been shown that for a sufficiently large number of mixture components, full-covariance GMMs have a higher recognition accuracy than diagonal-covariance GMMs, for an approximately equal amount of parameters. As the amount of parameters is used as an indication of recognition speed, this indicates that higher recognition accuracy can be achieved using full-covariance GMMs without sacrificing recognition speed.
- It has been shown that context-dependent phoneme-modelling, without the inclusion of language models and grammar, does significantly increase the phoneme-recognition accuracy compared to context-independent phoneme modelling.

1.6 Overview

This section provides a high-level summary of the work done in this thesis. The background and components of an automatic speech recognition system was sketched in section 1.2. The thesis can be summarised in the following three subsections:

1.6.1 The theory of statistical pattern recognition

In Chapters 2-4 the theory of statistical pattern recognition is explained in detail. The explanation focuses on three components of an automatic speech recogniser, namely the acoustic processor (Chapter 2), the acoustic models (Chapter 3 and 4) and the hypothesis search (Chapter 3).

Chapter 2 begins with a motivation of the necessity of processing speech before speech recognition can take place. This is followed by a discussion of the theory of the techniques used in the acoustic processor (or front-end). These techniques have been divided into three groups, namely *signal preprocessing*, *feature-extraction* and *feature post-processing techniques*. The signal-preprocessing techniques discussed are preemphasis and power normalisation. The feature-extraction techniques discussed are LPCCs, MFCCs and PLPs, while the feature post-processing techniques include unity-variance normalisation, cepstral mean subtraction, dynamic features and dimension reduction.

The acoustic models are statistically modelled using the theory of hidden Markov models, which is discussed in detail in Chapter 3. Firstly, the concept of hidden Markov models is introduced. This is followed by a discussion of the three standard HMM algorithms and the different types of HMMs. The chapter is concluded by explaining how the theory of HMMs is applied to the problem of automatic speech recognition.

Chapter 4 introduces the concept of modelling phonemes in the phonetic context in which it appears in an utterance. At first the different types of speech units that are typically modelled (i.e. words, syllables, phonemes) are discussed. Each type of unit is evaluated based on the following three criteria: accuracy, trainability and generalisability. The conclusion is made that phonemes are a good choice for a speech unit, if they are modelled in context, but that there are problems associated with the modelling of context. The problems are firstly a lack of enough training data and, secondly, the construction of models for phonetic contexts that do not occur in the training data. These problems can be overcome by the sharing of the parameters of different phoneme models. The parameter sharing technique that was chosen for this research is decision tree-based state clustering. The rest of the chapter is used to explain the theory and implementation of decision tree-based state clustering.

1.6.2 Implementation of a phoneme recogniser

The purpose of this thesis is to construct and evaluate phoneme models of South African English. The theory needed to understand the statistical modelling of both context-dependent and context-independent phonemes have been discussed. Chapter 5 explains how this theory is applied to construct phoneme models from a speech database.

The chapter starts by discussing the need for time-aligned or complete transcriptions

of a speech database. If such transcriptions do not exist, it is useful if they are created, as phonemes created from complete transcriptions generally perform better that those that are created from incomplete transcriptions. The construction of the complete transcriptions is discussed next. The rest of the chapter revolves around the procedure that is followed to construct the phoneme models. The first part is concerned with the construction of context-independent models, and the second part with the construction of context-dependent models.

1.6.3 Phoneme-recognition results

The last part of the thesis reports on the experimental findings of the phoneme recognition experiments. In order to evaluate and test the phoneme-modelling techniques, the techniques are first applied to the NTIMIT speech corpus. There are two reasons for this: The first reason is to confirm that techniques are implemented correctly by testing it on a database which has been extensively used. The second reason is to obtain baseline results to which the phoneme recognition experiments on South African English can be compared.

In Chapter 6 isolated phoneme recognition experiments are used to evaluate the signal processing and statistical modelling techniques. An optimal set of techniques is chosen, which is then used to perform isolated and continuous phoneme recognition on NTIMIT using both context-independent and context-dependent phoneme models (without using a language or grammar model). It was found that the inclusion of context, without the use of a language model or grammar, significantly increases the best recognition accuracy of context-dependent phoneme compared to context-independent phonemes.

Finally, the phoneme modelling techniques are applied to the South African English database. Continuous phoneme recognition experiments are performed using both context-independent and context-dependent phoneme models, without using a language or grammar model. The results follow the same trend as the baseline results, which indicates that the techniques are applicable to South African English, and that the South African English database is suitable for the development of continuous speech recognition applications.

Chapter 2

Signal processing

The processing of speech signals forms part of the acoustic processor (or front-end) of most modern speech recognisers. This chapter starts by briefly explaining why it is necessary to process the sampled speech signal $s(n)$, and then continues with a discussion on signal processing techniques. The signal processing techniques are divided into three main groups, namely preprocessing, feature extraction and feature post-processing. The purpose of this chapter is to describe the acoustic processor that has been used in the research documented in this thesis.

2.1 Motivation for speech processing

Speech is used to communicate information from the speaker to the listener. Speech starts with the speaker having an idea that he wishes to communicate. The speaker converts the idea into a linguistic structure (by choosing words and phrases to represent the idea), orders the phrases based on the grammatical rules of a particular language, and finally adds characteristics such as pitch, intonation or stress. By means of various neurological processes and muscle movements, this is then converted into an acoustic sound pressure wave, which is received by the listener's auditory system. The listener's auditory system processes this acoustic pressure wave and decodes it into the thought which the speaker wished to convey. The purpose of speech recognition is to replace the listener with an automatic speech recogniser. The mathematical formulation of speech recognition (Subsection 1.2.1) leads to the schematic diagram of a modern speech recogniser shown in Fig. 2.1. To understand the purpose of the signal processing techniques used in the acoustic processor, it is necessary to examine how a human being produces speech.

2.1.1 Acoustic theory of speech production

Fig. 2.2 shows a block diagram representation of human speech production. Engineers find it useful to think of speech production as an acoustic filtering process. The three

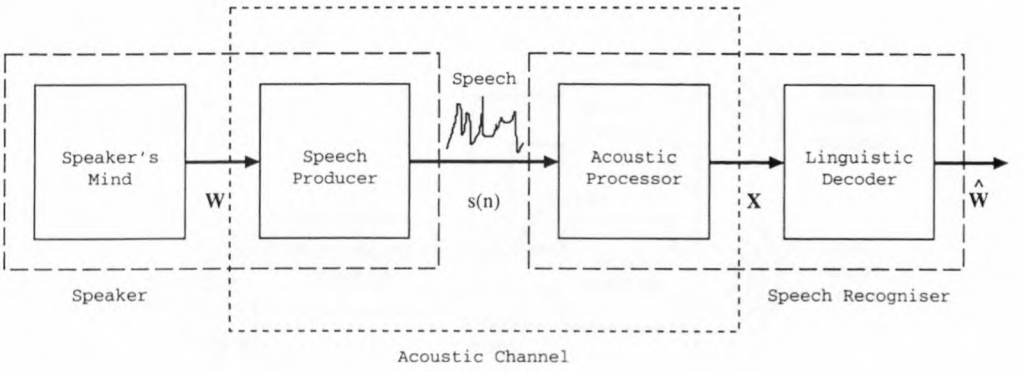

Figure 2.1: *The source-channel model of speech recognition* [23].

main cavities of the speech production system, i.e. the pharyngeal, nasal and oral cavity, form the main acoustic filter. The filter is excited by the organs below, and is loaded by a radiation impedance due to the lips. The articulators (vocal folds, velum, tongue, trachea), most of which form part of the filter itself, are used to change to properties of the system. A typical speech signal is shown in Fig. 2.3. The different types of speech in the signal can be clearly distinguished. If the spectral characteristics of the speech waveform is examined, it can be seen that the spectral characteristics vary in time (they are non-stationary), because the physical speech production changes over time. Speech can be divided into short sound segments, which possess similar acoustic properties. One of the most important properties of speech is that it is not a string of discrete well-formed sounds, but rather a series of "steady-state" or "target" sounds with intermediate transitions. The preceding and succeeding sound in a string can affect whether a target sound is reached completely, its duration, etc. This interplay is called *co-articulation*: the change in phoneme articulation and acoustics caused by the influence of another sound on the same utterance [9].

If the magnitude spectrum of one of the short sound segments is examined, various resonances in the frequency domain are noted. These resonances are caused by various acoustical cavities and sub-cavities formed from within the vocal tract cavities by the articulators. Conversely, each vocal tract shape is characterised by a set of resonant frequencies, also referred to as *formants*. To completely characterise the human speech production system would require a set of partial differential equations that describe the physical principles of air propagation in the vocal system [38]. Although extensive research has been performed in search of a universal theory of speech production, such a theory has yet to emerge. Nevertheless, the following three separate areas of human speech production are generally modelled: the source excitation, the vocal tract shaping and the effect of speech radiation. It is assumed here, as the majority of modern speech modelling techniques do, that these components are linear and separable. These assumptions are practical necessities needed to facilitate computationally feasible techniques for speech

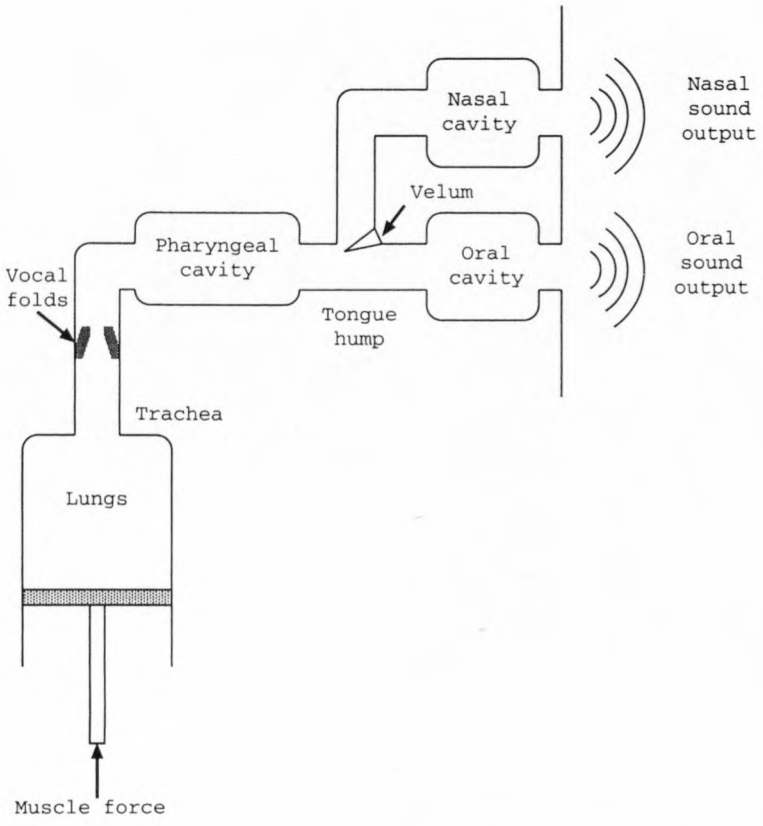

Figure 2.2: *A block diagram representation of human speech production* [9].

modelling.

It should be clear by now that a speech signal contains much more data than the idea which is being communicated from the speaker to the listener. In order to perform speaker-independent speech recognition, it is necessary to remove all the information pertaining to the speaker (such as the pitch and intonation) and the speech channel (e.g. noise etc.) in order to retain only the information needed to model the separate phonemes. To extract relevant discriminable information suitable to modern statistical pattern recognition techniques, it becomes necessary to perform some manner of processing on the speech signal $s(n)$. This chapter further discusses the signal processing which was used in this thesis.

2.2 Signal preprocessing

The purpose of signal preprocessing is to filter the speech signal so that only the information relevant to discriminating phonemes remains. For the purpose of speaker-independent speech recognition, the effects caused by speaker and channel identity need to be minimised.

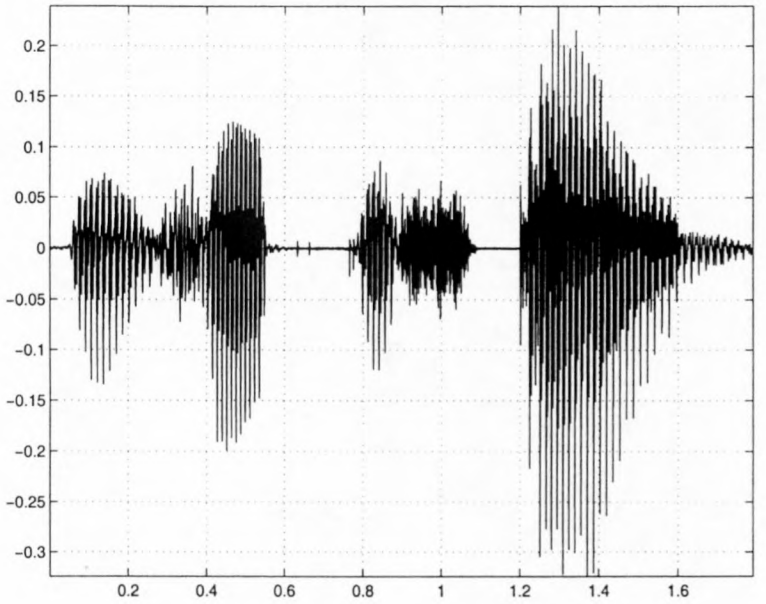

Figure 2.3: *The digitally sampled speech signal of "We hid the stain".*

2.2.1 Preemphasis

According to Deller [9] preemphasis involves the application of a filter that increases the relative energy of the high-frequency spectrum. This is done to suppress the spectral contributions of the larynx and the lips, and to spectrally flatten the signal. This also has the effect of making the signal less susceptible to numerical instability when performing linear predictive analysis. In the worst case, preemphasis will increase the influence of the higher formants on the spectrum. The preemphasis filter takes the form of

$$P_r(z) = 1 - \nu z^{-1}, \quad \text{where} \quad \nu \approx 1 \tag{2.1}$$

Typically, a value of $\nu = 0.98$ is used for the filter [9].

2.2.2 Power normalisation

Due to changing recording conditions, the average power in different speech recordings may differ. This will cause unwanted bias in the extracted feature vectors toward the signal with more power. This bias must be removed. By normalising the power in the speech signal to unity power it is ensured that the information in the different speech signals are weighed equally and that no bias results. Let $s(n)$ be a speech signal of length L. The power P_s of the speech signal $s(n)$ is calculated as:

$$P_s = \frac{1}{L} \sum_{n=1}^{L} s^2(n) \tag{2.2}$$

The power-normalised speech signal $s_{\text{pnorm}}(n)$ is then calculated as:

$$s_{\text{pnorm}}(n) = \frac{s(n)}{\sqrt{P_s}} \tag{2.3}$$

2.3 Feature extraction

2.3.1 Motivation

Having used preprocessing to suppress or remove information pertaining to the speaker or the channel, pattern recognition could be performed directly on the preprocessed signal. Early efforts in speech recognition did exactly that — a technique known as template matching. A few problems arise when using the preprocessed speech signal directly. The first problem is that different speech signals may differ in length. If one wishes to use a random process to represent the speech signal, the dimension of this random process would be determined by the length of the speech signal. Two problems arise from this:

- The random processes would have very large dimensions.
- The dimensions of the random processes would differ, making them difficult to compare.

In order to accurately estimate the parameters of random processes with large dimensions, a huge amount of data would be needed. Furthermore, the amount of computing time needed increases in the square of the dimension. If the dimension of the random vectors is doubled, the amount of computing time generally increases by four (this is known as the *curse of dimensionality* [6]). Techniques such as Dynamic Time Warping (DTW) were developed in an attempt to solve the problem of speech signals having different lengths. Modern speech recognisers use a technique called feature extraction. Feature extraction can be summarised by the following steps:

1. The speech signal is divided into a set of possible overlapping, equal-length analysis frames.
2. A set of parameters is extracted from each frame.
3. The parameters of each frame somehow characterises that frame.
4. The set of parameters can be represented by a vector of dimension D. The vector is referred to as a *feature vector*.

The feature extractor transforms the one-dimensional (preprocessed) speech signal $s(n)$ into the set of D-dimensional feature vectors $\mathbf{x}_1\mathbf{x}_2\ldots\mathbf{x}_T = \mathbf{X}$. Two main types of features are used in speech recognition. The one type is based on the *short-term Fourier spectrum*

of the signal and the other on *Linear Predictive Analysis* (LPA) of the speech signal. The question as to what is being characterised by the feature vectors remains unanswered. The features based on the Fourier spectrum characterises the frequency characteristics, while, in order to answer the question for LPA, it is necessary to discuss the way in which the speech production system is typically modelled.

2.3.2 Discrete-time filter model for speech production

The discrete-time filter model [38], shown in Fig. 2.4, is the speech production model used in this research. This model is called a *terminal-analogue* model, because the signals and systems involved are only superficially analogous to the true physical speech production system, except at the termini, where both systems produce similar speech waveforms. The vocal tract model $H(z)$ and radiation model $R(z)$ are excited by a discrete-time glottal excitation signal $u_{\text{glottis}}(n)$. During voiced speech the excitation uses an estimate of the local pitch period to set an impulse train generator that drives a glottal pulse shaping filter $G(z)$. During unvoiced speech the excitation source is a flat-spectrum noise source modelled by a random noise generator. Because speech is quasi-stationary (stationary for short periods of time), it would be possible to estimate the parameters of the filters $H(z)$, $R(z)$ and $G(z)$ for each of these stationary periods. The usual method of extracting these frames is to multiply the signal with a windowing function (such as the Hamming window). It is important to realise that the analysis frames can and do overlap. If the analysis frames are short enough, the assumption can be made that the speech is stationary within an analysis frame. The filter parameters of each frame could then be used to characterise that frame and could be used as feature vectors. The vocal tract

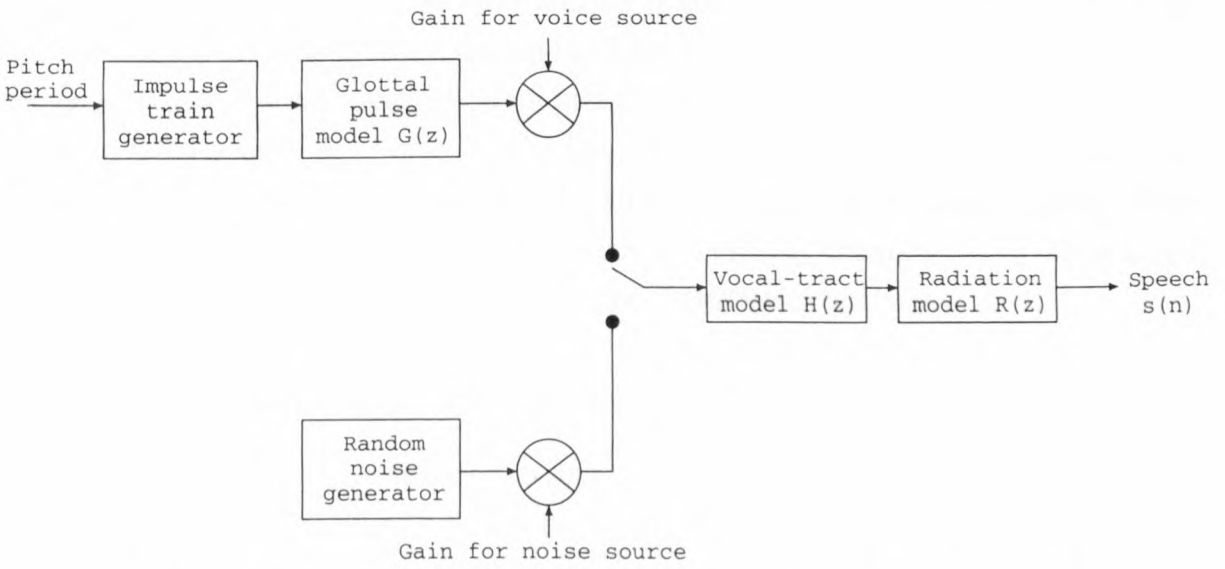

Figure 2.4: *The discrete-time filter model of speech production* [38].

filter $H(z)$ is modelled as

$$H(z) = \frac{H_0}{1 - \sum_{k=1}^{K} h_k z^{-k}} \tag{2.4}$$

while the radiation model $R(z)$ is modelled as

$$R(z) = 1 - z_0 z^{-1}, \quad z_0 \approx 1, \quad z_0 < 1 \tag{2.5}$$

For the purpose of speech recognition the glottal pulse shaping filter is less important, but it can be modelled using an all-zero filter. For an in-depth motivation for the modelling of the speech production filters, refer to Deller *et al.* [9]. The model of speech production can be simplified by grouping together the glottal, vocal tract and radation filters to form a single pole-zero function $\Theta(z)$, during a stationary frame of speech. This function takes the form

$$\Theta(z) = \Theta_0 \frac{1 + \sum_{i=1}^{O} b_i z^{-i}}{1 - \sum_{i=1}^{R} a_i z^{-i}} \tag{2.6}$$

which is driven by the excitation sequence

$$u(n) = \begin{cases} \sum_{q=-\infty}^{\infty} \delta(n - qP), \ q \in \mathbb{N} & \text{voiced speech} \\ \text{zero mean, unity variance,} & \\ \text{uncorrelated noise} & \text{unvoiced speech} \end{cases} \tag{2.7}$$

This simplification assumes that the filter $\Theta(z)$ changes with time. The simplified model is a decent approximation for most sounds, but fails on voiced fricatives. However, estimating the parameters of a pole-zero function is not a trivial task and therefore a further approximation is made by constraining the filter $\Theta(z)$ to be an all-pole function denoted by $\hat{\Theta}(z)$. There is some justification for using an all-pole filter, as a zero can be expressed with an infinity number of poles. Therefore, the all-pole approximation $\hat{\Theta}(z)$ will be an reasonable approximation to the pole-zero filter $\Theta(z)$, as long as a large number of poles are used. The method employed in this research to identify the parameters of the all-pole filter $\hat{\Theta}(z)$ of each analysis frame is called linear predictive analysis.

2.3.3 Linear predictive analysis

As mentioned previousy, the discrete-time speech production model $\Theta(z)$ (Eq. 2.6) is approximated with the all-pole function

$$\hat{\Theta}(z) = \frac{1}{1 - \sum_{i=1}^{M} \hat{a}_i z^{-i}} = \frac{1}{\hat{A}(z)} \tag{2.8}$$

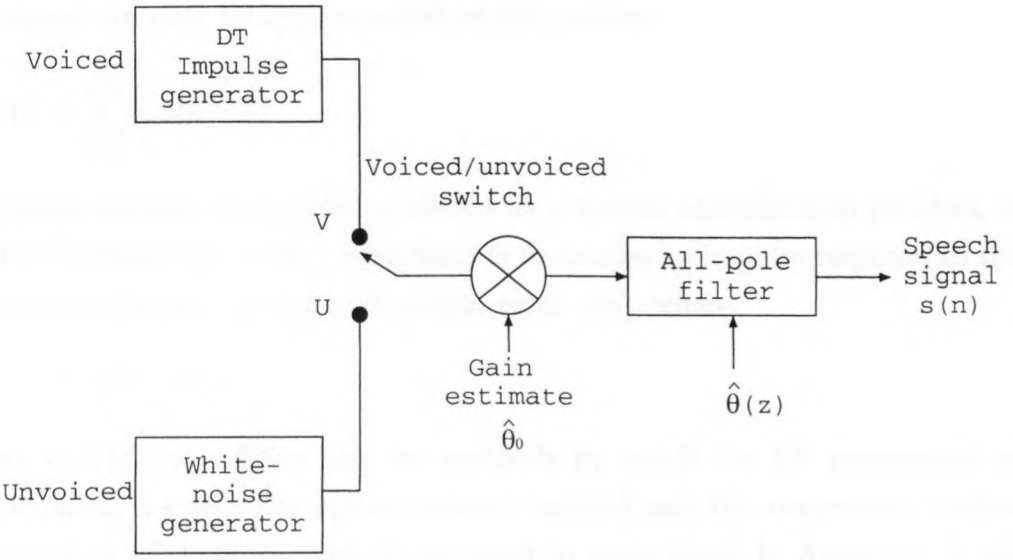

Figure 2.5: *The simplified, discrete-time filter model of speech production that is estimated using linear prediction analysis.*

The approximate model, which is referred to as the LP model, can exactly model the magnitude spectral characteristics of the speech (if M is taken as ∞), but not the phase characteristics. The use of the LP model is primarily a matter of analytical necessity, because the parameters of the all-pole model can be determined using simple *linear* equations. With regard to speech perception, the human ear is fundamentally "phase deaf" and therefore it is assumed that the information in the speech lies in the magnitude spectrum [9]. Fig. 2.5 illustrates the discrete-time speech production model to be estimated by using LP analysis. Using the LP model, the speech signal $s(n)$ can be modelled with the following equation:

$$s(n) = \sum_{i=1}^{M} \hat{a}_i s(n-i) + \hat{\Theta}_0 u(n) \tag{2.9}$$

which is shown figuratively in Fig. 2.6. Because the input signal $u(n)$ is unknown, the

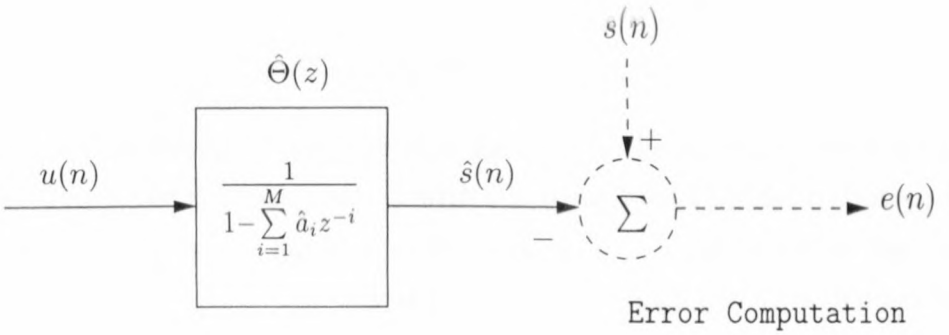

Figure 2.6: *System identification of the LP model of speech production.*

speech signal can only be approximated as $\hat{s}(n)$, where

$$\hat{s}(n) = \sum_{i=1}^{M} \hat{a}_i s(n-i) \tag{2.10}$$

If the feature extraction problem is viewed as a system identification problem, a classic method of deducing the model parameters is to ascribe an impulse response to the model that minimises the average squared output error $e(n)$, where

$$e(n) = s(n) - \hat{s}(n) \tag{2.11}$$

as shown in Fig. 2.6. There are two methods by which the LP parameters are typically calculated, namely the *autocorrelation* method and the *covariance* method. The implementation of these methods is discussed in more detail in Appendix A where the implementation of the different feature extraction algorithms is discussed.

Using LPA, it is possible to extract a finite set of M parameters which smooth the magnitude spectral characteristics of the speech signal $s(n)$. M is referred to as the order of the LP model. According to the speech production model, the speech is composed of an excitation signal $e(n)$ convolved with the impulse response of the vocal system model $\theta(n)$. The smoothing performed by the LP model does not resolve the vocal tract characteristics from the glottal dynamics. As the primary interest is in the vocal tract impulse response, it would be desirable to eliminate the influence of the excitation signal on the magnitude spectral characteristics. A solution to this problem is called cepstral analysis.

2.3.4 Cepstral analysis

The purpose of cepstral analysis is to separate the contribution of the excitation signal to the output speech signal, from the contribution of the vocal tract impulse response (which is what must be modelled). It would be relatively easy to suppress the contribution of the excitation signal by using a filter, if the following two conditions apply:

1. The speech signal $s(n)$ is a linear combination of the excitation signal $e(n)$ and the vocal tract impulse response $\theta(n)$, i.e. $s(n) = e(n) + \theta(n)$.

2. $e(n)$ and $\theta(n)$ are "separated" in the frequency domain.

According to the discrete-time speech production model, the speech signal $s(n)$ is a result of the convolution of the excitation signal with the vocal tract impulse response. Cepstral analysis transforms the speech signal $s(n)$ into the cepstral domain where the conditions mentioned above are met. The real cepstrum of a speech signal $s(n)$ is defined as

$$c_s(n) = \mathcal{F}^{-1}\left\{\log\left|\mathcal{F}\left\{s(n)\right\}\right|\right\} = \frac{1}{2\pi}\int_{-\pi}^{\pi} \log\left|S(\omega)\right| e^{j\omega n} d\omega \tag{2.12}$$

where $\mathcal{F}\{\cdot\}$ denotes the DTFT. Next, the influence of taking the logarithm of the DTFT of the speech signal is investigated. It is known that

$$s(n) = e(n) * \theta(n) \tag{2.13}$$

If the DTFT is taken on both sides of Eq. 2.13, the convolution in the time domain becomes multiplication in the frequency domain. Therefore the DTFT of $s(n)$ is

$$S(\omega) = E(\omega)\Theta(\omega) \tag{2.14}$$

By taking the logarithm of the magnitude spectrum of the speech signal, which is expressed as

$$\log|S(\omega)| = \log|E(\omega)| + \log|\Theta(\omega)| \tag{2.15}$$

the result is a linear combination of the logarithm of the magnitude spectra of the excitation signal and the vocal tract impulse response. The log magnitude spectrum is a new "frequency" domain and is referred to as the *quefrency* domain. By taking the IDTFT of the quefrency domain, the quefrency components are transformed to a new "time" domain, which is referred to as the cepstral domain. It should be clear that the cepstrum of the speech signal is a linear combination of the cepstra of the excitation signal and the vocal tract impulse response, or, expressed mathematically as

$$c_s(n) = c_e(n) + c_\theta(n) \tag{2.16}$$

Because the excitation signal varies much faster in time than the vocal tract impulse response, it can be assumed that they are well separated in the quefrency domain. The cepstrum of the vocal tract impulse response is found at the lower part of the quefrency domain while the cepstrum of the excitation signal is found at the higher parts of the quefrency domain. The separation of the two signals is referred to as "liftering" (which is analogous to filtering in the frequency domain). By applying a "low-time lifter", which amounts to taking only the first L samples of $c_s(n)$, an estimate of the cepstrum of the vocal tract impulse response $c_\theta(n)$ is obtained.

Feature extraction techniques based on combining both linear predictive analysis and cepstral analysis, have been found to achieve better speech recognition performance than techniques based on only LPA or CA. The three feature extraction techniques used in this research are based on short-time Fourier analysis, LPA and CA. Linear Predictive Coding Cepstral Coefficients (LPCCs) and Perceptual Linear Predictive Coefficients (PLPs) are based on LPA and CA, while Mel-Frequency Cepstral Coefficients (MFCCs) are based on short-time Fourier analysis and CA. The implementation of all these techniques is discussed in detail in Appendix A.

2.4 Feature post-processing

At this stage the one-dimensional speech signal $s(n)$ has been transformed into a set of D-dimensional feature vectors $\mathbf{X}$, which characterise the magnitude spectral characteristics of an overlapping set of analysis frames of the speech signal. Before this set of feature vectors can be used for speech recognition, it is necessary to perform some post-processing on the feature vectors. The reason for the post-processing is to improve the recognition accuracy of the subsequent recognition component.

2.4.1 Unity variance normalisation

One type of parameter that is estimated from the extracted feature vectors are covariance matrices. Due to the finite precision of computers, these estimated matrices can become ill-conditioned. It is therefore necessary to normalise the extracted feature vectors in order to improve the condition number of these covariance matrices. This is done by scaling each dimension of the features so that the standard deviation of that dimension is unity. The scaling needed for each dimension is computed from a set of speech training data.

2.4.2 Cepstral mean subtraction

Varying channel characteristics result in convolutional noise in the speech signal. This is transformed into additive noise in the cepstral domain, causing unwanted bias in the extracted features. In order to increase the robustness of the recogniser, it is necessary to remove this bias. A common method, called Cepstral Mean Subtraction (CMS), is to subtract the mean of each dimension of the cepstral features.

2.4.3 Dynamic features – velocity and acceleration

Dynamic features, more commonly known as *delta* features, are the temporal derivatives of the extracted feature vectors. Velocity features (Δs) are the first time derivatives, and acceleration features ($\Delta\Delta$s) are the second time derivatives. In statistical speech recognition it is usually assumed that the successive feature vectors are conditionaly independent. This is done in an attempt to make the recognition problem computationally tractable. The assumption is not very accurate, because the vocal apparatus forces continuity between successive spectral estimates of the speech signal [9]. The augmentation of the extracted or base feature vectors with velocity and acceleration features is done in an attempt to compensate for the errors that the conditionally independence assumption causes. It is also a means of distilling more information from the extracted feature vectors.

Simply using a difference equation is inappropriate to approximate the derivative, because it is very noisy for discrete signals. The deltas are therefore computed using a

polynomial approximation of the derivative. In this research, the following approximation was used to calculate the dynamic features:

$$\Delta\mathbf{x}(n) = -2\mathbf{x}(n-2) - \mathbf{x}(n-1) + \mathbf{x}(n+1) + 2\mathbf{x}(n+2) \tag{2.17}$$

of which the frequency response is shown in Fig. 2.7.

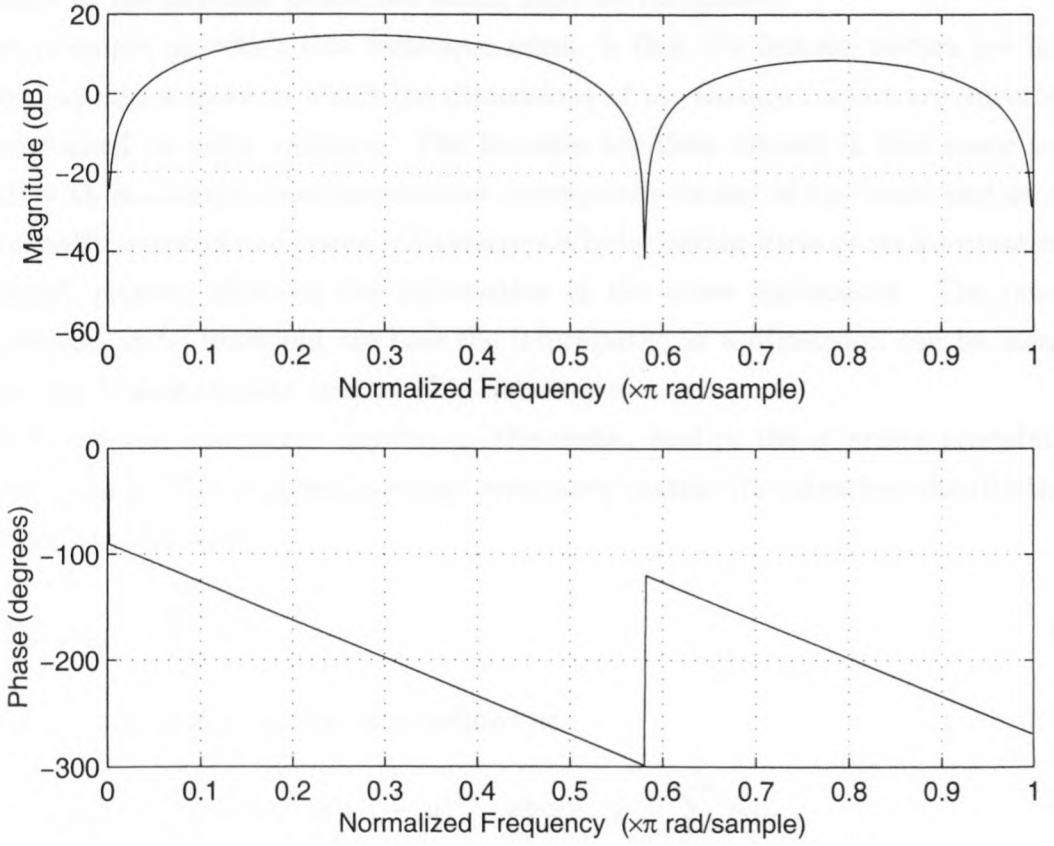

Figure 2.7: *The frequency response of the filter used to compute dynamic features such as velocity (Δs) and acceleration ($\Delta\Delta s$).*

Care must be taken at the endpoints of the speech signal to make provision for the fact that some of the feature vector samples do not exist.

2.4.4 Dimension reduction

By appending Δs, the dimension of the feature vectors is doubled, and by appending both Δs and $\Delta\Delta$s, the dimension of the feature vectors is tripled. This dramatically increases the amount of data needed to properly estimate the statistical parameters of the feature vectors. It is possible that, although the feature vectors span a P-dimensional space, the information only utilises a D-dimensional subspace in this P-dimensional space. This means that only D dimensions are needed to describe the information. It would be preferable to work in the lower-dimensional space, as the computational requirement is lower, and better use of the limited available data can be made.

Linear discriminant analysis, which is related to a variation of the Karhunen-Loeve transform [10, 25], is used to reduce the dimension of the feature vectors. This technique reduces the dimension through a linear transformation $\mathbf{K}$ of the feature vectors, while "maximising" the distance between different classes of data. The distance between classes is not actually changed – only the perspective of the data is changed. "Classes" in this case refers to the different phonemes which must be recognised.

The principle on which this technique relies, is that the feature vectors are linearly transformed into a space in which the dimensions of the feature vectors are uncorrelated and normalised to unity variance. The features are then rotated in this space so that the axis with maximum class information corresponds to one of the basis vectors of the dimensionally uncorrelated space. Dimensions which contain little or no information can be omitted without affecting the information in the other dimensions. The questions which remain to be answered are how the information in a dimension can be measured and how the transformation matrix $\mathbf{K}$ is determined.

Let $\mathbf{\Sigma}_i$ be the covariance matrix, μ_i the mean, and p_i the *a priori* probability of phoneme class i. The weighted average covariance matrix (or intraclass distribution) of the I phoneme classes is

$$\mathbf{\Sigma}_{\text{intraclass}} = \sum_{i=1}^{I} p_i \mathbf{\Sigma}_i \tag{2.18}$$

The interclass distribution matrix is defined as

$$\mathbf{\Sigma}_{\text{interclass}} = \sum_{i=1}^{I} p_i (\mu_i - \mu)(\mu_i - \mu)^T \quad \text{where} \quad \mu = \sum_{i=1}^{I} p_i \mu_i \tag{2.19}$$

Compute the eigenvector decomposition of the intraclass distribution matrix $\mathbf{\Sigma}_{\text{intraclass}}$. Let the eigenvalue matrix be $\mathbf{\Omega}_{\text{intraclass}}$ and the eigenvector matrix be $\mathbf{U}_{\text{intraclass}}$. Let $\mathcal{B}$ be the transformation matrix

$$\mathbf{Y} = \mathcal{B}^T \mathbf{X} \tag{2.20}$$

so that the dimensions of $\mathbf{Y}$ are uncorrelated. If the variance on each dimension is the same, therefore the features form a round distribution, rotation will not cause the dimensions of the features to become correlated again. The rotation matrix $\mathcal{B}$ can be calculated as

$$\mathcal{B} = \mathbf{U}_{\text{intraclass}} \mathbf{\Omega}_{\text{intraclass}}^{-\frac{1}{2}} \tag{2.21}$$

where the term $\mathbf{U}_{\text{intraclass}}$ uncorrelates the dimensions and the term $\mathbf{\Omega}_{\text{intraclass}}^{-\frac{1}{2}}$ scales the variance on each dimesion to unity (refer to [10] for more details). Transform the interclass distribution matrix $\mathbf{\Sigma}_{\text{interclass}}$ to the new dimensionally uncorrelated space

$$\mathbf{\Sigma}'_{\text{interclass}} = \mathcal{B}^T \mathbf{\Sigma}_{\text{interclass}} \mathcal{B} \tag{2.22}$$

Compute the eigenvector decomposition of the interclass distribution matrix $\mathbf{\Sigma}'_{\text{interclass}}$. Let the eigenvalue matrix be $\mathbf{\Omega}_{\text{interclass}}$ and the eigenvector matrix be $\mathbf{U}_{\text{interclass}}$. Let $\mathbf{U}'_{\text{interclass}}$ be the set of D eigenvectors corresponding to the D largest eigenvalues of $\mathbf{\Omega}_{\text{interclass}}$. Assuming that the dimension with maximum information corresponds to the dimension with the largest eigenvalue, the dimension of the feature vectors can be reduced by transforming the P-dimensional feature vectors $\mathbf{X}_P$ with the dimension reduction matrix

$$\mathbf{K} = \mathcal{B}\mathbf{U}'_{\text{interclass}} \tag{2.23}$$

The first transformation matrix $\mathcal{B}$ transforms the P-dimensional feature into a space in which the dimensions are uncorrelated. The second transformation matrix $\mathbf{U}'_{\text{interclass}}$ reduces the dimension of the feature vectors by rotating the dimensionally uncorrelated feature vectors onto a set of basis vectors, in which some of the dimensions contain little or no data. It can therefore be safely omitted. The D-dimensional feature vector $\mathbf{X}_D$ is consequently calculated as

$$\mathbf{X}_D = \mathbf{K}^T\mathbf{X}_P \tag{2.24}$$

For example, before applying LDA to the feature vectors, Fig. 2.8 shows the histogram of the cross-correlation coefficient magnitudes of 12-dimensional feature data modelled with full-covariance Gaussian pdfs. The y-axis represents the percentage of the total cross-correlation coefficients that falls within the range specified by the x-axis. It can be seen that the highest cross-correlation coefficient magnitude is approximately 0.8 and only approximately 40% of the cross-correlation coefficient magnitudes are less than 0.2. Fig. 2.9 shows the histogram of the cross-correlation coefficient magnitudes of the data after the application of LDA. The highest cross-correlation coefficient magnitude is approximately 0.55 and approximately 90% of the cross-correlation coefficient magnitudes are less than 0.2. By comparing Fig. 2.8 and Fig. 2.9 it can be seen that the application of LDA causes the dimensions of the feature vectors to be less correlated.

2.5 Summary

In this chapter, the motivations behind the use of an acoustic processor in an automatic speech recogniser were discussed. The components of the acoustic processor directly influence the performance of the subsequent speech recognition components, and therefore form a critical part of any speech recogniser. The acoustic theory of speech production was shortly discussed, and it was argued that speech contains much more data than the speaker-independent message that is being conveyed. It is therefore necessary to remove these speaker and channel artifacts by preprocessing the speech signal $s(n)$.

It was further argued that it is necessary to transform the one-dimensional speech signal into a set of D-dimensional feature vectors $\mathbf{X}$. This is done in an effort to make the

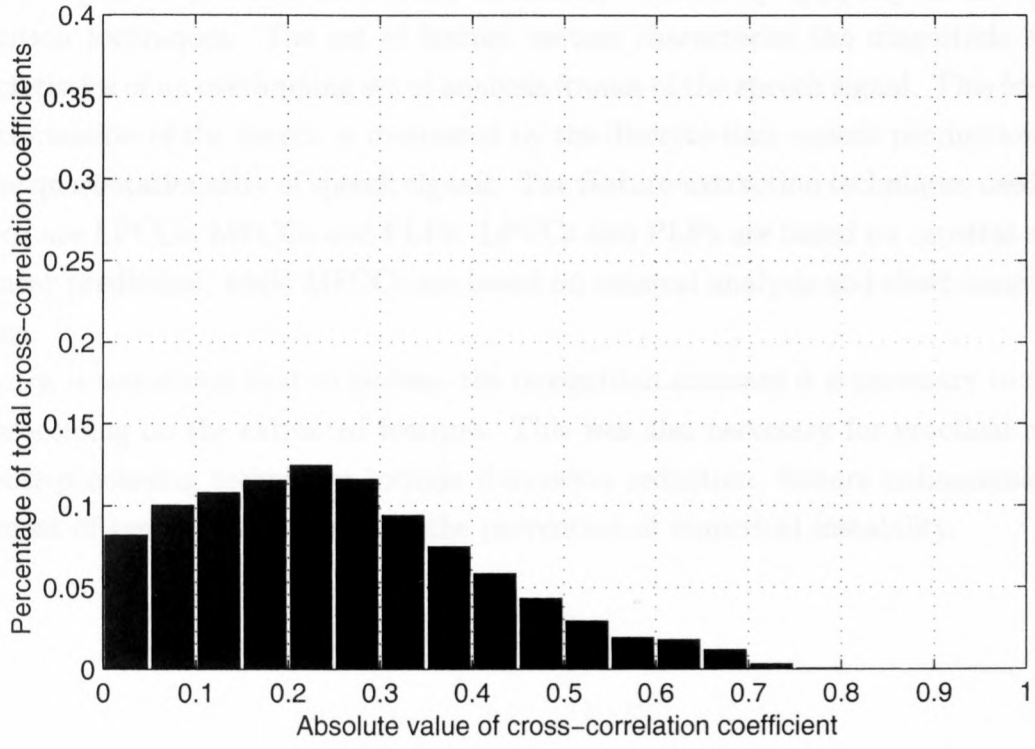

Figure 2.8: *A histogram of cross-correlation coefficient magnitudes of the feature vectors before the application of LDA.*

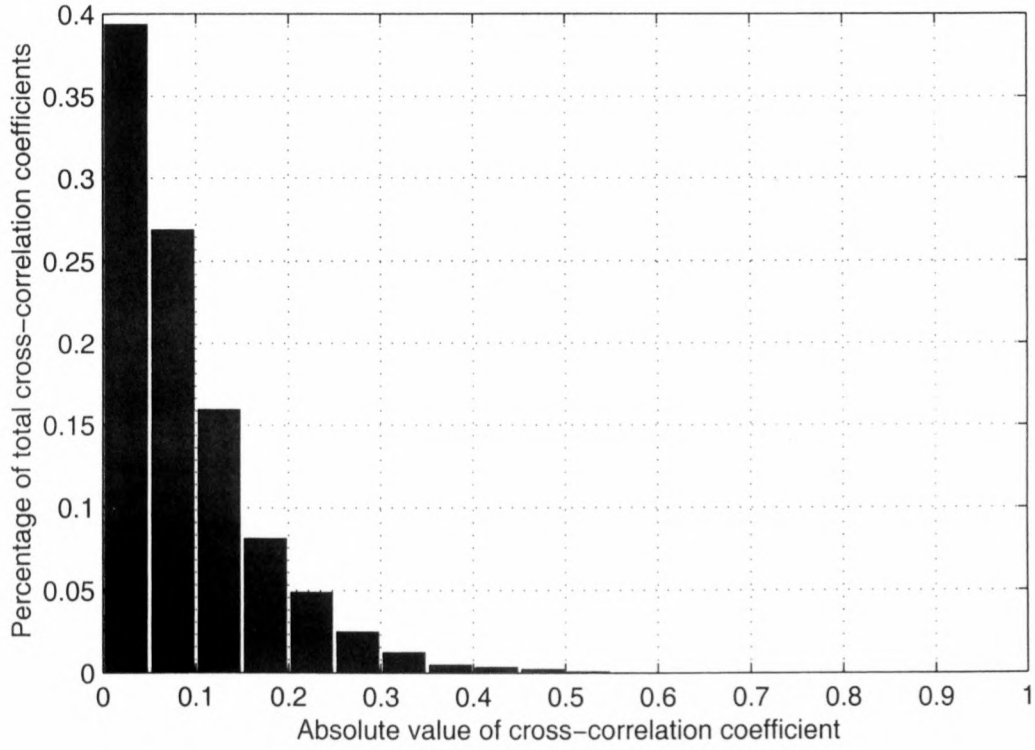

Figure 2.9: *A histogram of cross-correlation coefficient magnitudes of the feature vectors after the application of LDA.*

speech recognition problem more computationally feasible by applying modern pattern recognition techniques. The set of feature vectors characterise the magnitude spectral characteristics of an overlapping set of analysis frames of the speech signal. This frequency characterisation of the speech is motivated by the discrete-time speech production model and the quasi-stationarity of speech signals. The feature-extraction techniques used in this research are LPCCs, MFCCs and PLPs. LPCCs and PLPs are based on cepstral analysis and linear prediction, while MFCCs are based on cepstral analysis and short-time Fourier analysis.

Lastly, it was shown that to increase the recognition accuracy it is necessary to perform post-processing on the extracted features. This was also necessary for practical reasons. The post-processing techniques include dimension reduction, feature enhancement, improvement of system robustness and the prevention of numerical instability.

Chapter 3

Hidden Markov model theory

The previous chapter discussed the properties of the acoustic processor. This chapter will give an overview of the phonetic-acoustic models needed in order to determine the correct sequence of phonemes spoken. Because speech is quasi-stationary, it cannot be modelled with time-independent probability distribution functions (pdfs) such as Gaussian pdfs. A more suitable tool is the *Hidden Markov Model* (HMM), a statistical modelling technique which is capable of characterising the observed data samples of a discrete-time series.

This chapter starts by defining the HMM, the notation used to represent it and the assumptions inherent in HMMs. In order to create the acoustic models, it is necessary to know how the parameters of HMMs are estimated from the acoustic data (or feature vectors). In order to recognise speech it is necessary to determine the probability of a specific model. Next, some different types of HMMs is discussed, and lastly it is demonstrated how HMMs are applied when recognising sequences of phonemes.

The purpose of this chapter is to present the statistical theory from which the acoustic models are derived. The HMM material presented is necessary for the discussion on the modelling of context dependency. The HMM notation is used extensively during the presentation of the theory of decision tree based state clustering (discussed in Chapter 4). The HMM material is a combination of the presentations of HMM theory by the classic papers found in the literature [5, 9, 23, 26, 27, 35, 36, 37].

3.1 Definition and notation

The basic HMM theory was published in a series of papers by Baum and his colleagues [4] in the late 1960s and early 1970s. The underlying assumption is that the data samples of a discrete-time series can be characterised well by a parametric random process, and that the parameters of the stochastic process can be estimated in a precise and well-defined framework.

A HMM is a finite set of N states that is traversed according to a set of transition

probabilities. The transition probabilities describe the conditional probability of the HMM occupying a specific state, given a history of the states that were previously occupied. Each state has an associated output probability distribution, which defines the conditional probability that the HMM emits an observation symbol (or feature vector), given that the model is occupying a specific state. An HMM concurrently models two stochastic processes: the temporal structure and the locally stationary character of the system being modelled. The temporal structure is modelled by the transition probabilities and the locally stationary character is modelled by the output conditional pdf. Because only the sequence of observation symbols is known, the state sequence is said to be hidden (hence the name *hidden* Markov model). The HMM can be viewed as a doubly-embedded stochastic process with the underlying stochastic process (the state sequence) not directly observable.

The following notation is used for a discrete-observation HMM:

- T = length of the observation sequence.
- N = number of states in the model.
- L = average number of transition probabilitites per state.
- $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$, observation sequence (feature vectors).
- $\mathbf{Q} = \{q_1, q_2, \ldots, q_N\}$, states.
- $\mathbf{S} = \{s_1, s_2 \ldots, s_T\}$, hidden state sequence.
- $\mathbf{A} = [a_{ij}]$, $a_{ij} = P(s_t = q_j | s_{t-1} = q_i)$, $i, j = 1, 2, \ldots, N$, state probability transition matrix.
- $\mathbf{B} = \{b_j(\mathbf{x}_t)\}$, $b_j(\mathbf{x}_t) = f(\mathbf{x}_t | s_t = q_j)$, $j = 1, 2, \ldots, N$, the output probability distributions.
- $\pi = \{\pi_i\}$, $\pi_i = P(q_0 = s_i)$, the initial probabilities of the different states.

Fig. 3.1 illustrates a typical three-state, left-to-right HMM used to model phonemes. A complete specification of an HMM, Φ, includes three sets (matrices) of probability measures $\mathbf{A}, \mathbf{B}, \pi$. For convenience an HMM is represented as

$$\Phi = (\mathbf{A}, \mathbf{B}, \pi) \tag{3.1}$$

There is a mathematically equivalent representation of an HMM in which an output pdf is not associated with each state, but with each transition. The two equivalent representations are referred to as the "emission-on-state" and the "emission-on-transition" representations. For the purpose of phoneme modelling it was found to be more convenient

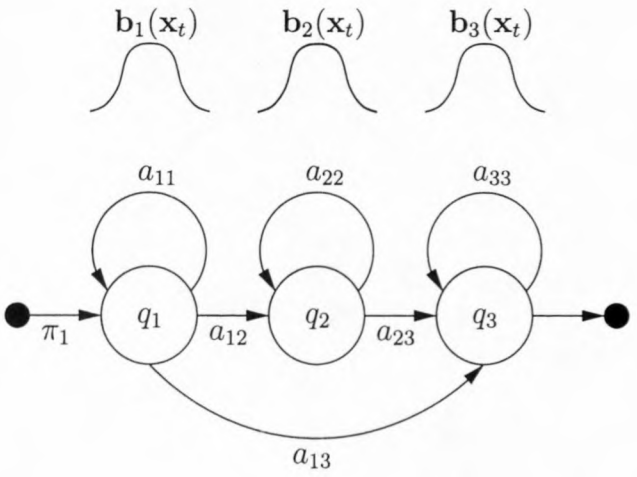

Figure 3.1: *A typical HMM used to model phonemes.*

to use the "emission-on-state" representation. In summary, a hidden Markov model is basically a Markov chain where the output observation is a random variable X generated according to an output probability distribution associated with each state.

3.2 HMM assumptions

Two assumptions are made for first-order hidden Markov models, namely the first-order *Markov assumption* and the *output-independence assumption.* The first-order Markov assumption is mathematically expressed as

$$P\left(s_t|\mathbf{x}_1^{t-1}, s_1^{t-1}\right) = P(s_t|s_{t-1}) \tag{3.2}$$

and states that the conditional probability of entering a state at time t is only dependent on the previous state and not the complete state sequence.

The output independence assumption is mathematically expressed as

$$P\left(\mathbf{x}_t|\mathbf{x}_1^{t-1}, s_1^{t}\right) = P(\mathbf{x}_t|s_t) \tag{3.3}$$

and states that the observation vector observed at time t is only dependent on the state that the HMM occupies at time t.

3.3 Standard HMM algorithms

Having defined the HMM, it is necessary to use HMMs to recognise both isolated and continuous speech. In order to recognise speech, HMM models need to be created in the first place, and thus it is necessary to estimate the parameters of the HMMs from a set of training data. These problems are referred to as the three basic problems of HMMs, which is stated as:

1. **The evaluation problem**: given a model Φ and a sequence of observations $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$, what is the probability $P(\mathbf{X}|\Phi)$, i.e. the probability that the model generates the observations?

2. **The decoding problem**: given a model Φ and a sequence of observations $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$, what is the most likely state sequence $\mathbf{S} = \{s_1 s_2 \ldots, s_T\}$ in the model that produces the observations?

3. **The learning problem**: given a model Φ and a set of observations, how should the model parameter $\hat{\Phi}$ be adjusted to maximise the joint likelihood $\prod_{\mathbf{X}} P(\mathbf{X}|\Phi)$?

By solving the evaluation problem, isolated phoneme recognition can be performed, since the likelihood $P(\mathbf{X}|\Phi)$ can be used to compute the posterior probability $P(\Phi|\mathbf{X})$. The HMM with the highest *posterior* probability can then be classified as the correct phoneme, given the observation sequence. This type of classification is referred to as maximum likelihood classification, since the model with the highest likelihood is regarded as the model which generated the observation sequence.

It is possible to build a larger HMM by concatenating a set of smaller HMMs. Phoneme-based word models could, for example, be created by concatenating the HMMs of the phonemes of which the word consists. By solving the decoding problem it can be determined what sequence of states (corresponding to a set of phonemes) is the most likely to have generated the observation sequence (or unknown speech). In this way, continuous speech can be recognised. The decoding of the hidden state sequence is also needed when solving the learning problem.

If the learning problem can be solved, the parameters of an HMM can automatically be estimated from an ensemble of training observations. In practice the learning problem is solved by adjusting an existing model's parameters Φ in such a way that the probability that the model generated the observation sequence is maximised.

3.3.1 Evaluating an HMM: forward algorithm

It is necessary to calculate the probability $P(\mathbf{X}|\Phi)$ of the T-length observation sequence $\mathbf{X} = (\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_T})$, given the HMM Φ. The simplest method is to enumerate all possible state sequences of length T that generate observation sequence $\mathbf{X}$, and then to sum all the probabilities:

$$P(\mathbf{X}|\Phi) = \sum_{\text{all } \mathbf{S}} P(\mathbf{S}|\Phi) P(\mathbf{X}|\mathbf{S}, \Phi) \tag{3.4}$$

Consider the specific state sequence $\mathbf{S} = \{s_1, s_2 \ldots, s_T\}$ of length T, where s_1 is the initial state. Using the *Markov* assumption, the state sequence probability in Eq. 3.4 can be

rewritten as

$$\begin{aligned} P(\mathbf{S}|\Phi) &= P(s_1|\Phi)\prod_{t=2}^{T} P(s_t|s_{t-1},\Phi) \\ &= \pi_{s_1} a_{s_1 s_2} \ldots a_{s_{T-1} s_T} \end{aligned} \tag{3.5}$$

For the same state sequence $\mathbf{S}$, the joint output probability along the path can be rewritten by using the *output independence* assumption:

$$\begin{aligned} P(\mathbf{X}|\mathbf{S},\Phi) &= P\left(\mathbf{x}_1^T|s_1^T,\Phi\right) \\ &= \prod_{t=1}^{T} P(\mathbf{x}_t|s_t,\Phi) \\ &= b_{s_1}(\mathbf{x}_1) b_{s_2}(\mathbf{x}_2) \ldots b_{s_T}(\mathbf{x}_T) \end{aligned} \tag{3.6}$$

Substituting Eqs. 3.5 and 3.6 into 3.4, the likelihood of the observation sequence given the HMM is obtained:

$$\begin{aligned} P(\mathbf{X}|\Phi) &= \sum_{\text{all } \mathbf{S}} P(\mathbf{S}|\Phi) P(\mathbf{X}|\mathbf{S},\Phi) \\ &= \sum_{\text{all } \mathbf{S}} \pi_{s_1} b_{s_1}(\mathbf{x}_1) a_{s_1 s_2} b_{s_2}(\mathbf{x}_2) \ldots a_{s_{T-1} s_T} b_{s_T}(\mathbf{x}_T) \end{aligned} \tag{3.7}$$

Eq. 3.7 can be interpreted as follows: for any given state sequence, start from the initial state s_1 with probability π_{s_1}. Take a transition from state s_{t-1} to state s_t and generate the observation $\mathbf{x}_t$ with probability $b_{s_t}(\mathbf{x}_t)$ until the last transition is reached.

Direct evaluation of Eq. 3.7 requires enumeration of N^T possible state sequences, which results in exponential computational complexity. To be precise, each calculation of likelihood would require $(2T-1)N^T$ multiplications and $N^T - 1$ additions. Fortunately, there exists a more efficient algorithm to calculate the likelihood. This procedure is called the *forward* algorithm. Seeing that the calculation of $P(s_t|s_{t-1},\Phi)P(\mathbf{x}_t|s_t,\Phi)$ involves only s_{t-1}, s_t and $\mathbf{x}_t$, it is possible to compute the likelihood $P(\mathbf{X}|\Phi)$ with recursion on t.

Define the forward probability $\alpha_t(i)$ as the probability that the HMM occupies state i at time t having generated partial observation $\mathbf{x}_1^t$, i.e.

$$\alpha_t(i) = P\left(\mathbf{x}_1^t, s_t = i|\Phi\right) \tag{3.8}$$

The forward probability $\alpha_t(i)$ can be solved inductively as follows:

1. **Initialisation**:

$$\alpha_1(i) = \pi_i b_i(\mathbf{x}_1), \quad 1 \leq i \leq N$$

2. **Induction**:

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}\right] b_j(\mathbf{x}_t), \quad 2 \leq t \leq T;\ 1 \leq j \leq N$$

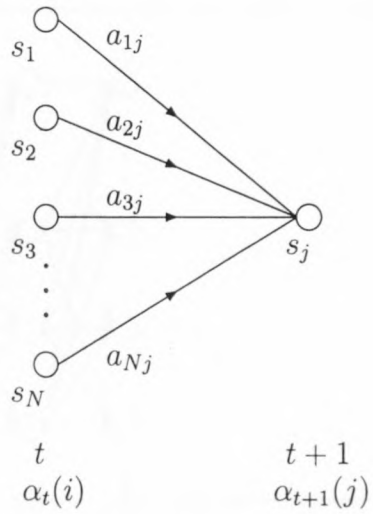

Figure 3.2: *An illustration of the computation of* $\alpha_{t+1}(j)$ [37].

3. **Termination**:

$$P(\mathbf{X}|\Phi) = \sum_{i=1}^{N} \alpha_T(i)$$

The induction step, which is the heart of the forward algorithm, is illustrated in Fig. 3.2. Using the *forward algorithm* the forward probabilities of state i at time t are calculated in a time-synchronous fashion. The forward probability is completely computed for each state i for time t before proceeding to time $t+1$. When the forward probabilies of the states at time T have been computed, the sum of all the forward probabilities at time T is the probability of generating the observation sequence $\mathbf{X}$. In speech applications, the HMM needs to end in some particular state or states, called the terminating states. The likelihood is the sum of the forward probabilities at time T of all the terminating states.

The complexity of the forward algorithm (for a fully connected model) is only $O(N^2T)$. To be precise, the forward algorithm requires $N(N+1)(T-1)+N$ multiplications and $N(N-1)(T-1)$ additions. The calculation of the forward probability is based on the trellis structure shown in Fig. 3.3. Because there are only N states, all possible state sequences must remerge into these N states, regardless of the length of the observation sequence.

3.3.2 Decoding an HMM: Viterbi algorithm

The forward algorithm computes the probability that a HMM generates an observation sequence by considering all paths (or state sequences) through the HMM. This is sufficient when performing isolated phoneme recognition, because an unclassified speech utterance, known to correspond to a phoneme, can be classified by computing the likelihood score for each phoneme model in a given set. In continuous speech recognition it is desirable to

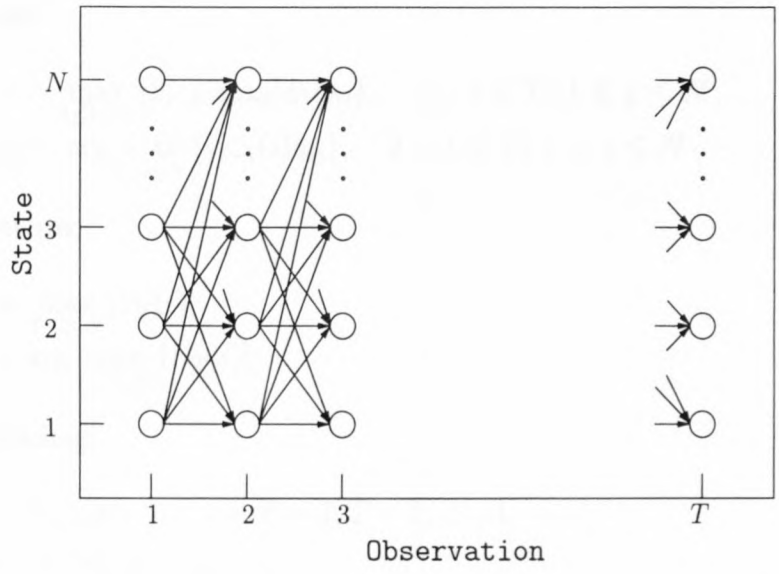

Figure 3.3: *The Forward-algorithm trellis structure* [37].

uncover the optimal hidden state sequence, because the HMM is usually built to consist of the concatenation of phonemes (or words). The hidden state sequence would then indicate the most likely sequence of phonemes that generated the observation sequence. Although the term "optimal" can be defined in a number of ways, in this research the criterion of "optimal" state sequence is the state sequence with the highest probability of being taken while generating the observation sequence. The purpose is to find the state sequence $\mathbf{S} = \{s_1, s_2 \ldots, s_T\}$ that maximises $P(\mathbf{S}|\mathbf{X}, \Phi)$, or equivalently maximises $P(\mathbf{S}, \mathbf{X}|\Phi)$. A formal technique based on dynamic programming, known as the *Viterbi* algorithm, can be used to find the most optimal state sequence.

The Viterbi algorithm can be shown to be dynamic programming applied to the HMM, or shown to be a modified forward algorithm. Instead of summing the probabilities at time t from different paths to the same destination state i, the Viterbi algorithm takes the maximum probability. Define the best-path probability $V_t(i)$ as the probability of the most likely state sequence at time t, which has generated the observation $\mathbf{x}_1^t$ and ends in state i, and thus

$$V_t(i) = \max_{s_1^{t-1}} P\left(\mathbf{x}_1^t, s_1^{t-1}, s_t = i|\Phi\right) \tag{3.9}$$

To be able to retrieve the optimal state sequence, it is necesssary to keep track of the preceding state of the best path for each time t and state s_i. This is done via the array $B_t(i)$. The best-path probability $V_t(i)$ can be solved inductively as follows:

1. **Initialisation**:

$$V_1(i) = \pi_i b_i(\mathbf{x}_1), \quad 1 \leq i \leq N$$
$$B_1(i) = 0$$

2. **Induction**:

$$V_t(j) = \max_{1 \leq i \leq N} [V_{t-1}(i)a_{ij}]\, b_j(\mathbf{x}_t), \quad 2 \leq t \leq T;\ 1 \leq j \leq N$$
$$B_t(j) = \arg \max_{1 \leq i \leq N} [V_{t-1}(i)a_{ij}], \quad 2 \leq t \leq T;\ 1 \leq j \leq N$$

3. **Termination**:

$$P^* = \max_{1 \leq i \leq N} [V_T(i)]$$
$$s_T^* = \arg \max_{1 \leq u \leq N} [B_T(i)]$$

4. **Backtracking**:

$$s_t^* = B_{t+1}(s_{t+1}^*), \quad t = T-1, T-2, \ldots, 1$$
$$\mathbf{S}^* = (s_1^*, s_2^*, \ldots, s_T^*)$$

where $\mathbf{S}^*$ denotes the optimal state sequence and $P^* \approx P(\mathbf{X}|\Phi)$ is an approximate of the probability that the observation sequence was generated by the HMM Φ. The order of the Viterbi algorithm is $O(N^2T)$.

3.3.3 Estimating HMM parameters: Baum-Welch algorithm

The last and most difficult problem to solve is the learning problem. The reason is that there is no known analytical method that maximises the joint probability of the observation sequences in a closed form. The *Baum-Welch* algorithm, which is based on the Expectation-Maximisation (EM) algorithm, is a method in which the parameters of an HMM can be iteratively adjusted so that the likelihood $P(\mathbf{X}|\Phi)$ is locally maximised. The HMM learning problem is a case of unsupervised learning in which the data is incomplete as a result of the hidden state sequence. Before the Baum-Welch algorithm can be discussed, it is necessary to define a number of probabilities.

3.3.3.1 Definition of auxiliary probabilities

Define the backward probability $\beta_t(i)$ as

$$\beta_t(i) = P\left(\mathbf{x}_{t+1}^T | s_t = i, \Phi\right) \tag{3.10}$$

where $\beta_t(i)$ is the probability of generating the partial observation $\mathbf{x}_{t+1}^T$ given that the HMM Φ is in state i at time t. Similarily to the forward probability, the backward probability can be inductively calculated as follows:

1. **Initialisation**:

$$\beta_T(i) = \frac{1}{N_F}, \quad 1 \leq i \leq N, N_F \text{ is the number of terminating states}$$

2. **Induction:**

$$\beta_t(i) = \left[\sum_{j=1}^{N} a_{ij} b_j(\mathbf{x}_{t+1}) \beta_{t+1}(j)\right], \quad t = T-1, \ldots, 1;\ 1 \leq i \leq N$$

Define $\gamma_t(i)$ as the probability of being in state i at time t, given the observation sequence $\mathbf{X}$ and the model Φ, i.e.

$$\begin{aligned} \gamma_t(i) &= P(s_t = i|\mathbf{X}, \Phi) \\ &= \frac{P(\mathbf{X}, s_t = i|\Phi)}{P(\mathbf{X}|\Phi)} \\ &= \frac{P(\mathbf{X}, s_t = i|\Phi)}{\sum_{k=1}^{N} P(\mathbf{X}, s_t = k|\Phi)} \end{aligned} \tag{3.11}$$

Since $P(\mathbf{X}, s_t = i|\Phi) = \alpha_t(i)\beta_t(i)$, Eq. 3.11 can be written as

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{k=1}^{N} \alpha_t(k)\beta_t(k)} \tag{3.12}$$

Define $\zeta_t(i, j)$ as the probability of taking the transition from state i to state j at time $t + 1$, given the model and observation sequence, i.e.

$$\begin{aligned} \zeta_t(i,j) &= P(s_t = i, s_{t+1} = j|\mathbf{X}, \Phi) \\ &= \frac{P(s_t = i, s_{t+1} = j, \mathbf{X}|\Phi)}{P(\mathbf{X}|\Phi)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(\mathbf{x}_{t+1}) \beta_{t+1}(j)}{\sum_{k=1}^{N} \alpha_T(k)} \end{aligned} \tag{3.13}$$

The paths that satisfy the conditions of Eq. 3.13 are illustrated in Fig. 3.4.

3.3.3.2 Iterative maximisation of the HMM parameters Φ

Using the EM algorithm, the parameters of the HMM $\Phi = (\mathbf{A}, \mathbf{B}, \pi)$ can be iteratively refined by maximising the likelihood $P(\mathbf{X}|\Phi)$ for each iteration. Denote $\hat{\Phi}$ as the new set of HMM parameters derived from the HMM parameters Φ of the previous iteration. The maximisation (using constrained optimisation techniques) of Baum's auxiliary function $Q(\Phi, \hat{\Phi})$ over $\hat{\Phi}$ is equivalent to maximising $P(\mathbf{X}|\hat{\Phi})$, because

$$Q(\Phi, \hat{\Phi}) \geq Q(\Phi, \Phi) \Rightarrow P(\mathbf{X}|\hat{\Phi}) \geq P(\mathbf{X}|\Phi) \tag{3.14}$$

Baum defined his auxiliary function as

$$\begin{aligned} Q(\Phi, \hat{\Phi}) &= \sum_{\text{all } \mathbf{S}} P(\mathbf{S}|\mathbf{X}, \Phi) \log P(\mathbf{X}, \mathbf{S}|\hat{\Phi}) \\ &= \sum_{\text{all } \mathbf{S}} \frac{P(\mathbf{S}, \mathbf{X}|\Phi)}{P(\mathbf{X}|\Phi)} \log P(\mathbf{S}, \mathbf{X}|\hat{\Phi}) \end{aligned} \tag{3.15}$$

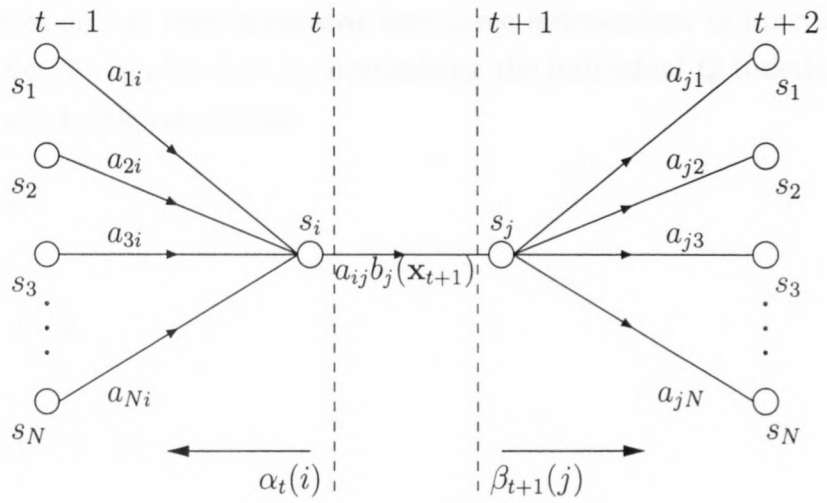

Figure 3.4: *An illustration of computation of the forward probabilities $\alpha_t(i)$ and the backward probabilities $\beta_{t+1}(j)$.*

where $P(\mathbf{X}, \mathbf{S}|\Phi)$ and $\log P(\mathbf{X}, \mathbf{S}|\hat{\Phi})$ can be expressed as:

$$P(\mathbf{X}, \mathbf{S}|\Phi) = \pi_{s_0} \prod_{t=1}^{T} a_{s_{t-1}s_t} b_{s_t}(\mathbf{x}_t) \tag{3.16}$$

$$\log P(\mathbf{X}, \mathbf{S}|\hat{\Phi}) = \log \hat{\pi}_{s_0} + \sum_{t=1}^{T} \log \hat{a}_{s_{t-1}s_t} + \sum_{t=1}^{T} \log \hat{b}_{s_t}(\mathbf{x}_t) \tag{3.17}$$

By using Eqs. 3.16 and 3.17, Eq. 3.15 can be rewritten as

$$Q(\Phi, \hat{\Phi}) = Q_\pi(\Phi, \hat{\pi}) + \sum_{i=1}^{N} Q_{\mathbf{a}_i}(\Phi, \hat{\mathbf{a}}_i) + \sum_{j=1}^{N} Q_{\mathbf{b}_j}(\Phi, \hat{\mathbf{b}}_j) \tag{3.18}$$

where

$$\begin{aligned} &\pi = [\pi_1, \pi_2, \ldots, \pi_N], \\ &\mathbf{a}_i = [a_{i1}, a_{i2}, \ldots, a_{iN}], \\ &\mathbf{b}_j \text{ is the parameter vector that defines } b_i(\cdot) \end{aligned} \tag{3.19}$$

and

$$Q_\pi(\Phi, \hat{\pi}) = \sum_{i=1}^{N} \frac{P(\mathbf{X}, s_0 = i|\Phi)}{P(\mathbf{X}|\Phi)} \log \hat{\pi}_i \tag{3.20}$$

$$Q_{\mathbf{a}_i}(\Phi, \hat{\mathbf{a}}_i) = \sum_{j=1}^{N} \sum_{t=1}^{T} \frac{P(\mathbf{X}, s_{t-1} = i, s_t = j|\Phi)}{P(\mathbf{X}|\Phi)} \log \hat{a}_{ij} \tag{3.21}$$

$$Q_{\mathbf{b}_j}(\Phi, \hat{\mathbf{b}}_j) = \sum_{t=1}^{T} \frac{P(\mathbf{X}, s_t = j|\Phi)}{P(\mathbf{X}|\Phi)} \log \hat{b}_j(\mathbf{x}_t) \tag{3.22}$$

Since the Q function has been separated into three independent Q functions, the maximisation on $Q(\Phi, \hat{\Phi})$ can be done by maximising the individual Q functions separately, subject to the stochastic constraints:

$$\sum_{i=1}^{N} \pi_i = 1 \tag{3.23}$$

$$\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i \tag{3.24}$$

$$\int_{-\infty}^{\infty} b_j(\mathbf{x}_t) = 1 \quad \forall j \tag{3.25}$$

Upon examination of Eq. 3.18, it can be seen that the maximisation process consists of the independent maximisation of the initial probabilities π, the transition probabilities of each state a_{ij}, and the output pdfs of each state $b_j(\mathbf{x}_t)$. The initial probabilities are, however, only special cases of the transition probabilities. It can therefore be seen that the Baum-Welch algorithm amounts to the independent maximisation of the Q functions associated with the two stochastic processes being modelled by the HMM, i.e. the hidden state sequence and the output observation sequence. When deriving the reestimation formulae for each of the sets of parameters, it will be seen that the reestimation formulae amounts to the Maximum Likelihood Estimate (MLE) of the parameters of the pdfs associated with each state. Although the transition probabilities (and initial probabilites) was defined in a sense as being "separate entities", it is important to realise that the transition probabilities for each state are in actuality only a discrete pdf.

According to the EM algorithm, the Baum-Welch (or forward-backward) algorithm guarantees a monotonic likelihood improvement on each iteration, and eventually the likelihood converges on a local maximum. The Baum-Welch algorithm can be described by the following steps:

1. **Initialisation**: choose an initial HMM estimate Φ.
2. **E-step**: compute the auxiliary function $Q(\Phi, \hat{\Phi})$ based on Φ.
3. **M-step**: compute $\hat{\Phi}$ according to the reestimation formulae to maximise the Q function.
4. **Iteration**: set $\Phi = \hat{\Phi}$, repeat from Step 2 until convergence.

3.3.3.3 Derivation of reestimation formulae

The separate auxiliary functions (Eqs. 3.20, 3.21 and 3.22) are all of the form

$$\sum_{j=1}^{N} w_j \log y_j \tag{3.26}$$

which can be shown (using Lagrange multipliers) to attain a global maximum at the single point

$$y_j = \frac{w_j}{\sum_{i=1}^{N} w_i}, \quad j = 1, 2, \ldots, N \tag{3.27}$$

Using Eqs. 3.27, 3.11, 3.12 and 3.13, the new estimates for the initial and transition probabilities can be written as:

$$\begin{aligned}
\hat{\pi}_i &= \frac{\frac{1}{P(\mathbf{X}|\Phi)} P(\mathbf{X}, s_0 = i|\Phi)}{\frac{1}{P(\mathbf{X}|\Phi)} \sum_{k=1}^{N} P(\mathbf{X}, s_0 = k|\Phi)} \\
&= \frac{P(\mathbf{X}, s_0 = i|\Phi)}{\sum_{k=1}^{N} P(\mathbf{X}, s_0 = k|\Phi)} \\
&= \gamma_0(i)
\end{aligned} \tag{3.28}$$

$$\begin{aligned}
\hat{a}_{ij} &= \frac{\frac{1}{P(\mathbf{X}|\Phi)} \sum_{t=1}^{T} P(\mathbf{X}, s_{t-1} = i, s_t = j|\Phi)}{\frac{1}{P(\mathbf{X}|\Phi)} \sum_{j=1}^{N} \sum_{t=1}^{T} P(\mathbf{X}, s_{t-1} = i, s_t = j|\Phi)} \\
&= \frac{\sum_{t=1}^{T} P(\mathbf{X}, s_{t-1} = i, s_t = j|\Phi)}{\sum_{t=1}^{T} P(\mathbf{X}, s_{t-1} = i|\Phi)} \\
&= \frac{\sum_{t=1}^{T} \zeta_{t-1}(i, j)}{\sum_{t=1}^{T} \gamma_{t-1}(i)}
\end{aligned} \tag{3.29}$$

The new estimates for the output pdfs depend on the type of pdf used. If, for instance, the output pdf of each state is a continuous Gaussian mixture distribution, i.e.

$$b_j(\mathbf{x}_t) = \sum_{k=1}^{\mathcal{M}} c_{jk} \mathcal{N}(\mathbf{x}_t, \mu_{jk}, \mathbf{\Sigma}_{jk}) \quad \text{where} \quad \sum_{k=1}^{\mathcal{M}} c_{jk} = 1 \tag{3.30}$$

The parameters of the output pdf $b_j(\mathbf{x}_t)$ are now the mixture weights c_{jk}, the means μ_{jk}, and the covariances $\mathbf{\Sigma}_{jk}$. For each of the parameters a reestimation formula needs to be derived from its Q function. The formulae are

$$\hat{c}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j, k)}{\sum_{t=1}^{T} \sum_{k=1}^{\mathcal{M}} \gamma_t(j, k)} \tag{3.31}$$

$$\hat{\mu}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)\mathbf{x}_t}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{3.32}$$

$$\hat{\mathbf{\Sigma}}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot (\mathbf{x}_t - \mu_{jk})(\mathbf{x}_t - \mu_{jk})'}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{3.33}$$

where $\gamma_t(j,k)$ denotes the probability of being in state j at time t with the kth mixture component accounting for $\mathbf{x}_t$, i.e.

$$\begin{aligned} \gamma_t(j,k) &= \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}\right]\left[\frac{c_{jk}\mathcal{N}(\mathbf{x}_t, \mu_{jk}, \mathbf{\Sigma}_{jk})}{\sum_{m=1}^{M} c_{jm}\mathcal{N}(\mathbf{x}_t, \mu_{jm}, \mathbf{\Sigma}_{jm})}\right] \\ &= \gamma_t(j)\left[\frac{c_{jk}\mathcal{N}(\mathbf{x}_t, \mu_{jk}, \mathbf{\Sigma}_{jk})}{b_j(\mathbf{x}_t)}\right] \end{aligned} \tag{3.34}$$

3.3.3.4 Multiple observation sequences

The Baum-Welch algorithm discussed above is based on one training observation sequence. It can easily be generalised to multiple observation sequences under the assumption that the sequences are independent of each other. To train an HMM from E data sequences is equivalent to finding the HMM parameter Φ that maximises the joint probability

$$\prod_{e=1}^{E} P(\mathbf{X}_i|\Phi) \tag{3.35}$$

The training procedure performs the Baum-Welch algorithm on each independent observation sequence to calculate the expectations in the reestimation formulae (which are sometimes referred to as counts). The counts are then added across the E data sequences respectively and finally the model parameters are normalised to make them sum to one. This constitutes one iteration of the Baum-Welch reestimation; iteration continues until convergence. For example, the extended reestimation formula for the transition probabilities can be written as:

$$\hat{a}_{ij} = \frac{\sum_{e=1}^{E}\sum_{t=1}^{T^e} \zeta_{t-1}^{e}(i,j)}{\sum_{e=1}^{E}\sum_{t=1}^{T^e} \gamma_{t-1}^{e}(i,j)} \tag{3.36}$$

where $\gamma_{t-1}^{e}(i)$ and $\zeta_{t-1}^{e}(i,j)$ respectively denote the $\gamma_{t-1}(i)$ and $\zeta_{t-1}(i,j)$ from the e^{th} data sequence and T^e denotes the corresponding length.

3.4 Types of HMMs

Using the algorithms of the previous section one can perform isolated and continuous speech recognition and, from a set of training data, estimate the parameters of a given HMM. In theory, the reestimation algorithms should give values for the HMM parameters that correspond to the local maximum of the likelihood function. The assumption is made that the HMM parameters are specified by the problem itself. The question remains how the initial estimates and structure of the HMM are chosen so that the local likelihood maximum is as close to the global maximum as possible. Unfortunately, there is no theoretically sound method of estimating both the *transition structure* (or topology) and the statistical parameters of the HMM. All that can be done is to use knowledge of the situation to design a HMM structure, and then to use the above algorithms to reestimate the model parameters.

This section will discuss some of the different types of HMMs that are used when modelling speech. The different aspects of HMMs that are discussed include the topology of the HMM, the state output pdf and higher-order HMMs.

3.4.1 Topology

The topology of an HMM is the structure of the transition matrix $\mathbf{A}$. This determines the way in which an HMM can move from one state to another. This sections therefore deals with the manner in which an HMM models the temporal characteristics of a speech signal. The two most popular HMM topologies used for modelling speech are the fully-connected and the left-to-right (Bakis) topology.

3.4.1.1 Fully connected HMM

In an HMM with a fully-connected topology, every state of the model can be reached, in a single step, from every other state. The fully connected HMM is a special case of the ergodic HMM. All other first-order topologies are special instances of the fully-connected topology, but with some of the transition probabilities equal to zero. Because all other first-order topologies are special instances of the fully-connected topology, it means that if an HMM technique is applicable to fully-connected topologies, it is applicable to all first-order topologies. Fig. 3.5 illustrates a three-state fully-connected HMM.

3.4.1.2 Left-to-right HMM

The left-to-right (or Bakis) model was first proposed for the modelling of speech by R. Baker [2]. The underlying state sequence associated with the left-to-right model has the property that, as time increases, the state index stays the same or increases. Thus, the

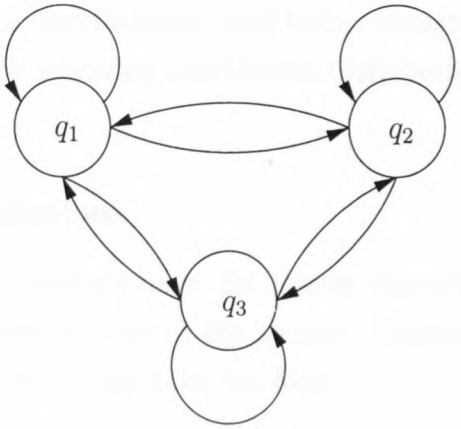

Figure 3.5: *A three-state fully-connected HMM.*

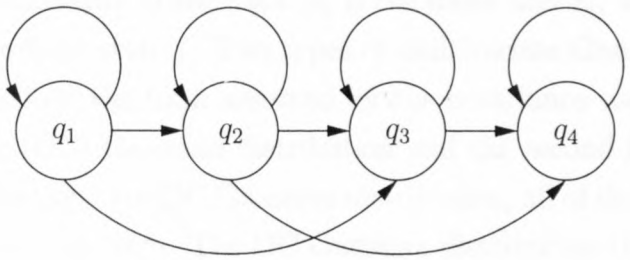

Figure 3.6: *A four-state left-to-right HMM.*

HMM states can only proceed from left to right. The left-to-right HMM is illustrated in Fig. 3.6 and has the property of being able to model signals of which the properties change over time in a successive manner (such as speech). This fundamental property of left-to-right HMMs is described by the transition probabilities having the property

$$a_{ij} = 0, \quad j < i \tag{3.37}$$

and the initial state probabilities having the property

$$\pi_i = \begin{cases} 0, & i \neq 1 \\ 1, & i = 1 \end{cases} \tag{3.38}$$

This means that the state sequence must start in state 1 and end in state N.

3.4.2 State output probability distributions

This section deals with probability distributions used to model the locally stationary character of a speech signal. The complexity of first-order HMMs is typically determined by state output pdfs. The pdfs must be specific enough to allow discrimination between different phonemes, as well as robust enough to allow for the expected variability inherent in natural speech. Although discrete probability distributions are much less computationally intensive than continuous pdfs, comparative results have shown that speech recognition

systems based on continuous distributions yield higher recognition accuracies [32]. Therefore, in this thesis, only the following continuous distributions have been used as state output pdfs:

3.4.2.1 Gaussian distributions

Multivariate Gaussian distributions have the lowest entropy and thus make the fewest assumptions about the characteristics of the speech. Gaussian distributions are characterised by second-order statistics and take the form

$$b_j(\mathbf{x}_t) = (2\pi)^{-\frac{D}{2}} |\mathbf{\Sigma}_j|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}_t-\mu_j)'\mathbf{\Sigma}_j^{-1}(\mathbf{x}_t-\mu_j)} \tag{3.39}$$

where D is the dimensionality of the data, μ_j is the mean and $\mathbf{\Sigma}_j$ is the covariance of the Gaussian distribution from state j. Two types of multivariate Gaussian distributions are commonly used, based on the form assumed by the covariance matrix. The first is the *Diagonal-Covariance* (DC) Gaussian distribution and the second is the *Full-Covariance* (FC) Gaussian distribution. For DC Gaussian distribution, all of the off-diagonal elements of the covariance matrix are zero. The DC Gaussian distribution therefore requires much less data to properly estimate its parameters than the FC Gaussian distribution, but assumes that the individual components of the observation vector are uncorrelated and thus statistically independent.

3.4.2.2 Mixture Gaussian distributions

Because speech rarely forms a single "hump" in the feature space, the use of single Gaussian distributions does not accurately model the shape of speech characteristics in the feature space. Mixture Gaussian distributions attempt to model the speech characteristics more accurately by using a weighted sum of Gaussian distributions. The $\mathcal{M}$-mixture Gaussian distribution of state j has the form:

$$b_j(\mathbf{x}_t) = \sum_{m=1}^{\mathcal{M}} c_{jm} b_{jm}(\mathbf{x}_t), \text{ where} \tag{3.40}$$

$$b_{jm}(\mathbf{x}_t) = (2\pi)^{-\frac{D}{2}} |\mathbf{\Sigma}_{mj}|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}_t-\mu_{mj})'\mathbf{\Sigma}_{mj}^{-1}(\mathbf{x}_t-\mu_{mj})}, \text{ and} \tag{3.41}$$

$$\sum_{m=1}^{\mathcal{M}} c_{jm} = 1 \tag{3.42}$$

The parameters of a single Gaussian distribution can be estimated with only a single pass of a MLE algorithm, but mixture Gaussian distributions require multiple iterations of the expectation-maximisation algorithm to obtain a proper estimate of the model parameters.

3.4.2.3 Tied-mixture distributions

The tied-mixture distribution has a similar form as the mixture Gaussian distribution, except that all states share the same bank of distribution functions. Each state has its own set of distribution weights. The tied-mixture Gaussian distribution of state j has the form:

$$b_j(\mathbf{x}_t) = \sum_{m=1}^{\mathcal{M}} c_{jm} b_m(\mathbf{x}_t), \text{ where} \tag{3.43}$$

$$b_m(\mathbf{x}_t) = (2\pi)^{-\frac{D}{2}} |\mathbf{\Sigma}_m|^{-1} e^{-\frac{1}{2}(\mathbf{x}_t - \mu_m)' \mathbf{\Sigma}_m^{-1} (\mathbf{x}_t - \mu_m)}, \text{ and} \tag{3.44}$$

$$\sum_{m=1}^{\mathcal{M}} c_{jm} = 1 \tag{3.45}$$

3.4.2.4 Shared state distributions

The final type of distribution is based on the concept of parameter tying. An equivalence relation is constructed between the HMM parameters of different states, and thus the number of independent parameters in the model is reduced. Parameter tying is used when the state output pdf is known to be the same (or nearly the same) in two or more states. Shared state distributions are especially useful when there is insufficient training data to reliably estimate the number of model parameters. Shared (or tied) state distributions are extensively used in the technique used to model context-dependent phonemes, which is discussed in Chapter 4.

3.4.3 Time duration modelling

The previous two sections have shown in what manner the temporal and locally stationary characteristics of speech are modelled. However, first-order HMMs do not adequately represent the temporal nature of speech. This is because the probability of state occupancy decreases exponentially in time, i.e.

$$d_i(t) = a_{ii}^t (1 - a_{ii}) \tag{3.46}$$

One method of improving the standard HMM is to explicitly model the state time duration [28, 40]. This is achieved by replacing the self-transition probability of a state with an explicit duration probability distribution. At time t, the process enters the state i with duration τ with discrete probability distribution $d_i(\tau)$, during which the observations $\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \ldots, \mathbf{x}_{t+\tau}$ are generated. The HMM occupies state j with transition probability a_{ij} only after the appropriate τ observations have been generated in state i. By setting the time duration pdf to be the exponential pdf of Eq. 3.46, the time duration HMM is equivalent to the standard HMM without explicit time duration modelling. J.D. Ferguson

[13] proposed the Ferguson model in which state time duration is modelled by adding extra states in order to model the time duration.

One of the drawbacks of explicit time duration modelling is the increase in computational complexity by a factor of $O(D^2)$, where D is the time duration distribution length. Another problem is the large number of extra parameters that need to be estimated, which could be alleviated by using a continuous, parametric state duration distribution instead of the discrete pdf $d_i(\tau)$.

3.4.4 Higher-order HMMs

The discussions of the previous sections have been assumed to be applicable to HMMs that comply with the first-order Markov assumption. Another method of improving the inadequate duration modelling of HMMs is to relax the first-order Markov assumption and to make the underlying state sequence a second- or higher-order Markov chain. For a K-order HMM, the transition probability between two states at time t would depend on the states which the process occupied at time $t-1$, $t-2$, ..., and $t-K$. Traditionally, higher-order HMMs have been approached by extending the Viterbi and forward-backward algorithms to take the higher order Markov assumption into account:

$$P(s_t|\mathbf{x}_1^{t-1}, s_1^{t-1}) = P(s_t|s_{t-1}, s_{t-2}, \ldots, s_{t-K}) \tag{3.47}$$

This requires the re-implementation of existing algorithms in order to use the higher-order HMMs. J.A. du Preez demonstrated that for any higher-order HMM, a mathematically equivalent first-order HMM exists, which makes it possible to process any higher-order HMMs using known first-order algorithms [12]. Although conventional higher-order HMMs offer more accurate modelling capabilities than first-order HMMs, they are computationally very expensive as the number of transition probabilities increase exponentially with the order. When duration modelling is modelled using higher-order HMMs that are represented by its first-order equivalents, the number of transition probabilities increase linearly with the order.

3.5 Applying HMMs to speech recognition

The theoretical foundation of HMMs that was laid in the previous sections is applicable to the solving of general pattern recognition problems using HMMs. The question which begs to be asked is how these techniques are applied when attempting to automatically recognise speech. The recognition of speech is divided into two problems, namely the recognition of words spoken in isolation, and the recognition of a sequence of words in continuous (or natural) speech. Solving the isolated speech recognition problem is the easier of the two. Assume that a vocabulary of $\mathcal{V}$ phonemes must be recognised. Each

of these phonemes ρ is modelled by a distinct HMM Φ_ρ. The HMM is constructed from a set of K spoken utterances of the phoneme. The phoneme utterances in the training set are not spoken in isolation, but occur in continuous speech. It is also necessary to be able to distinguish between phonemes and silence. Therefore, a silence model needs to be constructed.

3.5.1 Isolated-speech Recognition

When performing isolated phoneme recognition it is known that the unknown speech signal corresponds to a *single* phoneme, and it only remains to be determined *which* phoneme it is. The unknown signal is processed with the signal processing techniques of Chapter 2 to extract a set of feature vectors $\mathbf{X}$. For each of the phoneme HMMs, the likelihood $P(\mathbf{X}|\Phi_\rho)$ that the model generated the set of feature vectors is computed. The phoneme ρ^* whose model has the highest likelihood is selected as the correct phoneme, i.e.

$$\rho^* = \arg \max_{1 \leq \rho \leq \mathcal{V}} [P(\mathbf{X}|\Phi_\rho)] \tag{3.48}$$

A mathematically equivalent method of performing isolated phoneme recognition is embedding the phoneme models in a parallel-HMM (illustrated in Fig. 3.7). Determine the optimal path through the parallel-HMM for each unknown speech signal, which is equivalent to calculating the phoneme model with the highest likelihood. Let k be the number of correctly classified phonemes and let n be the total number of phonemes tested. The recognition accuracy $\text{ACC}_{\text{isolated}}$ of the isolated recognition system is determined simply as

$$\text{ACC}_{\text{isolated}} = \frac{k}{n} \tag{3.49}$$

3.5.2 Continuous-speech recognition

Continuous-speech recognition is considerably more difficult than isolated speech recognition. The reason for this is that the unknown speech signal corresponds to an unknown length *sequence* of phonemes of which the phoneme boundaries are not known. It is therefore necessary to consider the possibility of each phoneme starting at any arbitrary time frame. Continuous phoneme recognition can be solved by the standard HMM formulation by embedding the phoneme HMM into a spotter-HMM (illustrated in Fig. 3.8). The spotter-HMM has an *initial* and *collector* null state. From the initial state, the HMM can make a transition to any of the first states of any of the phoneme HMMs. The final state of each phoneme HMM is linked to the collector state. In order to enable a sequence of phonemes to be recognised, the collector state is linked to the initial state. The optimal path through the spotter-HMM corresponds to the most optimal sequence of phonemes

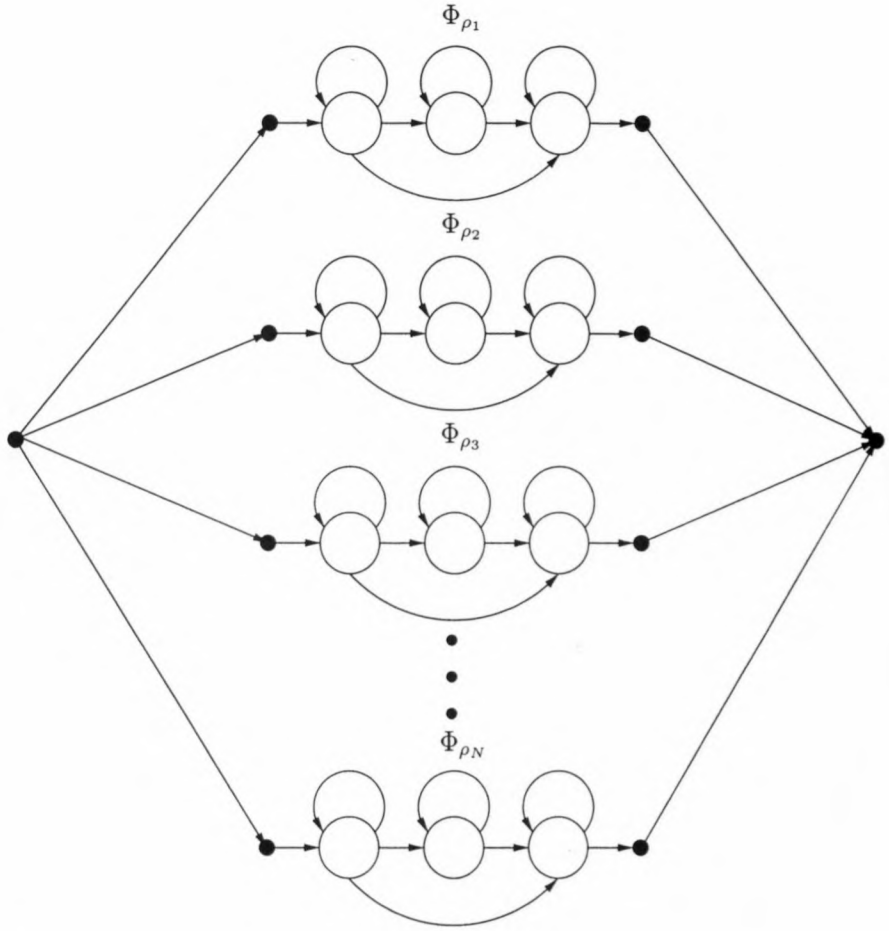

Figure 3.7: *A parallel-HMM used for isolated phoneme recognition of N phonemes.*

and can be determined using the Viterbi algorithm. Althought it is possible to use a unigram language model with the spotter HMM topology, no language model (a zero-gram model) is used. By embedding the phoneme HMM into an ergodic HMM (where each state corresponds to a phoneme HMM) it is possible to incorporate a bigram language model. The incorporation of language models into continuous-speech recognition, and the efficient decoding of continuous speech, forms a seperate field of research. Because this thesis is primarily concerned with the modelling of phonemes and because the phoneme models constructed as a result of this thesis will be incorporated into a larger dialogue system, these aspects will not be discussed. Refer to [18] for a very comprehensive discussion of these topics.

Determining the accuracy of a continuous speech recognition system is also not as simple as is the case with isolated speech recogntion systems. Three types of errors occur during continuous speech recognition:

- **Substitution** (SUB): an incorrect phoneme is substituted for a correct phoneme.
- **Deletion** (DEL): a correct phoneme is omitted in the recognised sequence.

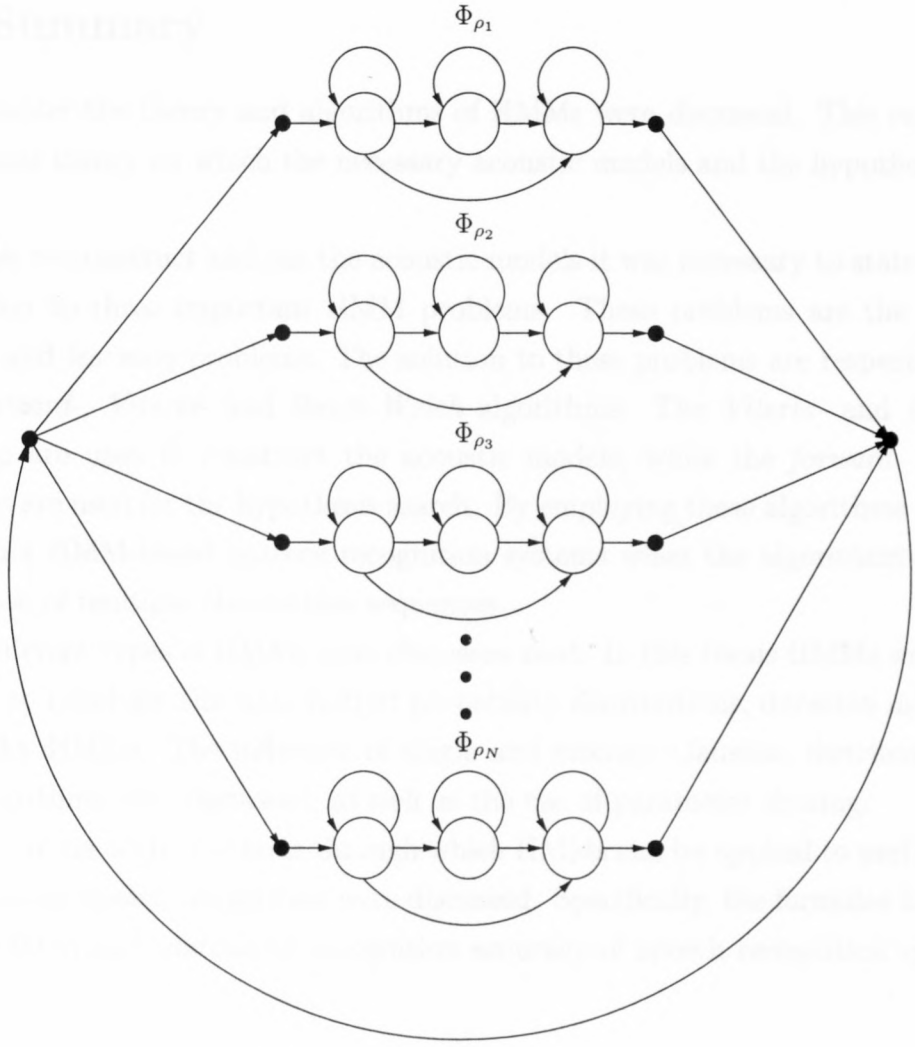

Figure 3.8: *A spotter-HMM used for continuous phoneme recognition of sequences which can contain any combination of N phonemes.*

- **Insertion** (INS): a phoneme is added into the recognised sequence.

The errors are determined by aligning the recognised and correct phoneme sequences using dynamic programming algorithms. If NUM is the number of phonemes in the correct sequence, the Phoneme Error Rate (PER) of a continuous speech recogniser can be determined using the following equation:

$$\text{PER} = \frac{\text{SUB} + \text{DEL} + \text{INS}}{\text{NUM}} \tag{3.50}$$

The accuracy ($\text{ACC}_{\text{continuous}}$) of the recogniser is then determined as:

$$\text{ACC}_{\text{continuous}} = 1 - \text{PER}. \tag{3.51}$$

3.6 Summary

In this chapter the theory and algorithms of HMMs were discussed. This represents the foundational theory on which the necessary acoustic models and the hypothesis search is built.

In order to construct and use the acoustic models it was necessary to state and discuss the solution to three important HMM problems. These problems are the *evaluation-*, *decoding-* and *learning* problems. The solution to these problems are respectively known as the *forward-*, *Viterbi-* and *Baum-Welch* algorithms. The *Viterbi-* and *Baum-Welch* algorithms are used to construct the acoustic models, while the *forward-* and *Viterbi* algorithms are used for the hypothesis search. By employing these algorithms it is possible to construct HMM-based pattern recognition systems when the algorithms are adapted to make use of multiple observation sequences.

The different types of HMMs were discussed next. In this thesis HMMs were classified according to topology, the state output probability distributions, duration modelling and higher-order HMMs. The influence of single and mixture Gaussian distributions on the HMM algorithms were discussed, as well as the use of parameter sharing.

Finally, some of the methods through which HMMs can be applied to perform isolated and continuous speech recognition were discussed. Specifically, the formulae for determining the isolated and continuous recognition accuracy of speech recognition systems were explained.

Chapter 4

Modelling context dependency

In the previous two chapters the foundation of statistical modelling techniques was laid. The foundation is divided into two main concepts. Firstly, it was discussed why it is necessary to extract a finite set of salient features from the one-dimensional speech signal. Secondly, statistical pattern recognition techniques, particularly the theory of hidden Markov models, were explained. The material in those two chapters enables one to build a system that is applicable to many pattern recognition problems. If a speech recogniser was built, with some constraints and based on the previous two chapters, it would have reasonable recognition performance. It is possible to increase this performance by examining and exploiting aspects peculiar to speech, such as context variability. This chapter focuses on the modelling of phonemes in their different allophonic contexts in order to increase recognition accuracy.

4.1 Background and motivation

> "The pattern recognition techniques used for speech recognition rely on the invariance of the data represented by the particular model. In continuous speech every instance of a particular speech unit can be different. Some of these differences are random, but a great deal can be accounted for by consistent *contextual* effects. In order to improve recognition accuracy it is necessary to take these consistent variations into account, while still allowing for the random variations always present in speech" – Odell [31].

Speech recognition systems model speech units, which might seem to be obvious. What is less obvious is what type of speech units can be modelled appropriately. Three selection criteria need to be balanced when choosing appropriate modelling units:

1. **Accuracy**: the units should be able to accurately represent the acoustic realisations of units in different contexts.

2. **Trainability**: there should be enough data to estimate reliable parameters of the acoustic unit.

3. **Generalisability**: one should be able to derive any new word from the predefined set of acoustic units.

4.1.1 Whole-word models

The most natural speech units are whole words, and whole-word models have been used extensively in many recognition systems. A distinct advantage is that whole-word models capture the phonetic co-articulation inherent within these words. For small-vocabulary speech recognition, whole-word models are appropriate speech units, but when whole-word models are scaled to large-vocabulary speech recognition, a few problems arise. For general-purpose large-vocabulary speech recognition, it is difficult to build whole-word models because:

- Every task contains new words without any available training data.
- Each word needs to be treated individually and even single words can have different acoustic realisations. Therefore, speech data cannot be shared across word models and therefore implies that a prohibitively large amount of training data and storage is necessary in order to build context-dependent word models.
- It is very expensive to model inter-word co-articulation or to adapt a word-based system for a new speaker, channel or context usage.
- Per definition, whole-word models are language dependent and, for this reason, cannot be shared across different languages.

If whole-word models are measured by using the three criteria of appropriate speech units, it can be seen that whole-word models are very *accurate* – if there is enough training data. They are, however, only *trainable* for small vocabularies and are typically not *generalisable*.

4.1.2 Syllables

The next logical choice of unit is syllables. Syllables contain phone clusters that contain the most variable contextual effects. Syllables fill the gap between whole-word models and phones, but is subject to the same type of problem as whole-word models. For example, in the English language there are over 30 000 syllables. Another problem associated with syllables is that there is not general consensus about what a syllable is. According to [18], the syllable is a unit that has intuitive plausibility, but remains difficult to pin down

precisely. It would be possible to build whole-word models using syllables. Therefore syllables are *generalisable*. But due to the large number of syllables they are not easily *trainable* with current methods. Furthermore, there has not been much research done to conclusively determine the *accuracy* of syllables. These problems make syllabic acoustic units unsuitable to current pattern recognition techniques, although literature indicates that better speech recognition would result from syllable-based recognisers.

4.1.3 Phonetic models

Seeing that there is such a large number of words, would it not be possible to use a relatively small number of sub-word speech units as building blocks to build word models? Phones are a well-known set of sub-word models, which also has the advantage of being vocabulary and language independent. In the English language there exists about 50 phones, which can be reliably modelled with only a few hundred sentences. If phones are measured with the selection criteria, it can be seen that phones are *trainable* and *generalisable*. Unfortunately phones are not *accurate* enough, because the asuumption is made that a phone in any context is identical and are therefore *overgeneralised*. In reality one can find dramatic context variability at the phonetic level. The context dependence becomes even more pronounced in continuous speech, because the phonemes are not fully realised. Although speech is modelled as the concatenated sequence of independent phonemes, these phonemes are not produced independently. The vocal articulators move at relatively slow speeds and cannot move instantaneously from one position to another. Each phoneme is influenced not only by the position of the vocal articulators during its production, but also by their movements before and after production. In continuous speech the realisation of a phoneme is strongly influenced by its immediately neighbouring phonemes.

4.1.4 Context-dependent phonetic models

By modelling each phoneme *in the context in which it occurs*, it is possible to significantly improve the recognition accuracy. This unfortunately comes at a price: when all the phonetic context are enumerated the number of distinct models required increase by several orders of magnitude, which reintroduces the trainability problem of whole-word models. A reasonable size speech corpus will provide enough examples of all context-independent phonemes. Due to the uneven distribution of phones in context in general speech, even large speech corpora will provide only a few occurrences of some phonetic contexts (if at all). It therefore becomes necessary to balance the trainability and accuracy with parameter-sharing techniques to ensure that the available training data is used efficiently.

The amount of context which can be modelled varies. *Monophone* context implies that

a single model represents all contexts. *Biphone* context implies that each model represents a phone in a particular left or right context. Left context refers to the preceding phone, while right context refers to the following phone. *Triphone* context implies that a single model represents a phone with both left- and right context specified.

In this research phonemes have been chosen as modelling unit as they fulfill the trainability and generalisability criteria. By modelling the phonemes in context it is possible to fulfill the accuracy criterion as well.

4.2 Issues associated with context modelling

Since more context is specified for triphones than for biphones and monophones, the context modelling of biphones and monophones forms a subset of the context modelling of triphones. Any issue associated with the context modelling of triphones will exist to a lesser agree when modelling biphones or monophones. If an issue can be resolved for the modelling of triphones, it can be argued that it can be resolved for the modelling of acoustic units with lesser phonetic context as well. For the rest of this chapter the discussion regarding context-dependent phonemes will focus on triphones, but the techniques discussed will be just as applicable to context-dependent phonemes with lesser context. In this section the two main issues associated with context modelling, i.e. *trainability* and the *modelling of unseen contexts* will be examined in more detail.

4.2.1 Trainability

To maximise the performance of a hidden Markov model based speech recogniser it is necessary to strike a balance between the complexity of the models and the ability to reliably estimate the parameters from the available data. If their parameters have been reliably estimated, the best performance would be expected from the most complex models. Having a parameter that has only been estimated from a few examples leads to a degradation in the recognition accuracy. Typically, more than a hundred examples of each phoneme are required to reliably estimate a model parameter. As has been mentioned before, there are many triphone contexts which occur rarely in the training data (if at all). It is necessary to ensure that the triphone models are trainable in order to reliably estimate the parameters from the training data. The three main types of methods used to ensure trainability are:

- **Backing-off**: is the substitution of poorly estimated, more context specific models with better estimated, less context specific models. A poorly estimated triphone can be substituted with a better estimated biphone (or even monophone) model. The disadvantage of this method is that all the biphone and monophone models

need to be available before the construction of the triphone models can take place. Furthermore, if a large number of triphones are backed off, only a few models will actually have triphone context specified, which defeats the purpose of modelling triphones in the first place.

- **Parameter smoothing**: is the smoothing of the parameters of more context specific models with the parameters of less context specific models. Such techniques include deleted interpolation [23] as was used in the *SPHINX* and *SPHINX II* systems [26, 27]. This has the advantage of preserving the context dependency of the triphones, but increases the robustness of the parameters. There is also the added overhead of needing a validation set of data.

- **Parameter sharing**: is the explicit sharing of models or parts of models between different contexts [50]. Because many phones have similar effects on neighbouring phones it is desirable to determine these instances of similar contexts and merge them. This method ensures that the model parameters are reliably estimated whilst maintaining context dependency. Furthermore, it leads to a much more manageable number of models. Fig. 4.1 illustrates triphone HMMs in which the distributions are shared:

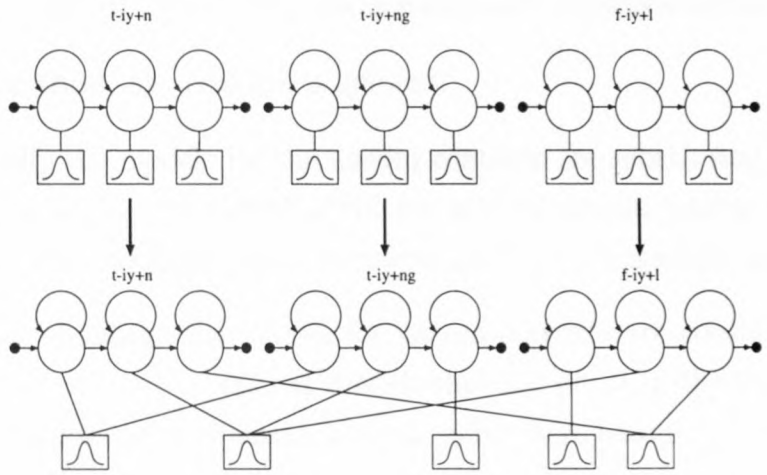

Figure 4.1: *The top row represents three triphone HMMs that do not share distributions. Each state has its own distribution (as indicated by the lines connecting states with distributions) for a total of nine distinct distributions. The bottom row represents the same three triphone HMMs, but parameter sharing has been applied. There are only five distinct distributions, which are shared between k^{th} states of each HMM (as indicated by the lines connecting states with distributions).*

From the discussion above one can see that parameter sharing is the most attractive of the three types of methods due to its preservation of context dependency and the

fact that it does not need a validation set. Parameter sharing methods include such techniques as *generalised triphones* [26] and *senones* [20]. Generalised triphones share the parameters of whole models, which can lead to the merging of two models when only parts of the model are similar. *Senones* are sub-phonetic units, which are actually a set of similar HMM states. Determining what parameters should be shared is done via a top-down or a bottom-up approach. The bottom-up approach assumes that all contexts are distinct and merges similar contexts to produce more trainable but less specific models. These approaches include agglomerate clustering techniques, which are purely data-driven [19, 20]. Top-down approaches assume all contexts are the same and are grouped. A splitting procedure is used to produce more specific and accurate models. These approaches include decision tree based state clustering [1, 31, 51, 52], which is data-driven but includes linguistic knowledge.

4.2.2 Phoneme modelling of unseen contexts

As mentioned previously, the uneven distribution of phones in context in general speech causes even large speech corpora to provide only some phonetic contexts. It is therefore possible to encounter context-dependent phonemes during recognition that did not appear in the training data. Contexts that did not appear in the training data are called "unseen contexts". Three ways of constructing phoneme models of unseen contexts are:

- **Ostrich method**: the problem is ignored.
- **Backing off**: the models for the unseen contexts are substituted with biphone or monophone models. The quality of the less context specific models is often not good enough and leads to degradation in search time and recognition performance.
- **Parameter sharing**: the models for unseen contexts are constructed by sharing parameters of context-dependent models that did occur in the training data. The bottom-up approach of parameter sharing suffers here because it is impossible to use it to construct models of unseen contexts. Top-down approaches such as decision trees can easily be used to construct models of unseen contexts by using linguistic knowledge about the unseen contexts (which will be explained in more detail later on).

Clearly the first method is not an adequate solution, while backing off models for unseen contexts do not lead to acceptable recognition rates for large vocabularies. Therefore, parameter sharing, specifically decision tree-based state clustering, is used to construct the models for unseen contexts.

Based on the previous discussions on trainability and the modelling of unseen contexts, decision tree-based state clustering is the method of choice used in this thesis. The

advantages of decision trees can be summarised as follows:

- The hierarchical structure and the linguistic questions will guarantee that the decision tree can be utilised to construct phoneme models of unseen contexts, where the constructed models have the same level of context dependency as the phoneme models that did occur in the training data.
- Expert knowledge is incorporated in the form of linguistic questions that are used to split each node of the tree and which are used to construct phoneme models of unseen contexts.
- The tree growing procedure can be constrained to ensure that similar contexts, which are clustered (referred to as leaf nodes) are only generated for contexts that have sufficient examples in the training data in order to reliably estimate model parameters.
- The decision tree can be used not only to construct triphone models, but is equally applicable to less (or more) context specific models.

In the next section the theory and construction of decision trees will be discussed.

4.3 Decision tree-based state clustering

Decision tree-based state clustering is the application of Classification And Regression Trees (CART) to the problem of determining which probability distributions should be shared. It has the advantage of elegantly solving the data insufficiency problem as well as being able to synthesize models of unseen contexts. It is a data-driven technique which also combines linguistic knowledge to determine the sharing of the parameters. The senonic decision trees are both state dependent and phone dependent. This means that only the distributions (called senones) of the set of allophones $\mathbb{C}_\rho$ derived from the same base phone ρ and from the same HMM state k, will be tied. Consequently, one decision tree will be constructed for each state k of each base phone ρ. The senonic decision tree is a binary tree used to classify target phones by asking binary questions in a hierarchical manner. The decision tree classifies Markov states of allophones occurring in the training corpus by asking linguistic questions of a set of predetermined simple categorical linguistic questions. Binary decision trees have become a very popular method of constructing context-dependent phonemes in the last few years. The discussion of the theory of binary decision trees was taken in part from several sources [1, 11, 19, 20, 23, 30, 31, 39, 51, 52]. First an example of decision tree-based clustering of fruit will be discussed to illustrates the concepts of decision tree-based clustering.

4.3.1 An example: decision tree-based fruit clustering

Imagine that a DSP company has been contracted to develop a fruit sorting machine for a fruit processing factory. The fruit sorting machine must be able to identify the fruit (specifically apples, pears, and bananas) as they move along on a conveyor belt in order to separate the fruit into different containers. The DSP company, which is based in some desert, sends someone down to the local fruit market to buy some fruit, but unfortunately only a limited number of fruit can be obtained, with no possibility of obtaining more fruit before the deadline of the project. Imagine that 100 apples, 105 pears and 90 bananas are available and that a minimum of 20 fruit is necessary in order to properly train a fruit model. Colour, shape and size will be used to train the recognition system, but because there are so few fruit available to train the recognition system, it is initially decided to make a single model for apples, another for pears and another for bananas.

However, it is found that the recognition performance based only on three fruit models is not good enough. It is decided to split the single model for each fruit into separate groups of models based on the colour and size. For example, instead of having one model for an apple, the company now has a group of nine models for apples i.e. small size red, green and yellow models, normal size red, green and yellow models, and large size red, green and yellow apple models. The 100 apples can be divided into 20 red, 50 green and 30 yellow apples. Unfortunately there are not enough fruit to obtain accurate separate models based on the colour and size of the fruit. Therefore, the separate models must share model parameters using the decision tree-based clustering technique. For instance to determine the parameter sharing for the group of apple models the following is done:

1. All 100 apples are placed in a single basket, which is called the "root" basket.

2. The apples in the root basket are split into two new baskets based on yes/no questions about the colour and size of the apples. How the apples are split is determined by choosing the question that maximises the difference between the apples in the two new baskets resulting from the split. Assume the best way to split the first basket is by putting all 50 green apples in basket 2 and all non-green apples (20 red and 30 yellow) in basket 3.

3. After this the best questions, that respectively maximises the apple splitting of basket 2 and basket 3 are determined. For instance, the green apples of basket 2 are split into 20 small apples and 30 non-small apples, and the red and yellow apples of basket 3 are split into 30 large apples and 20 non-large apples. Now it is decided whether splitting basket 2 or basket 3 maximises the difference between the apples in the two new baskets. This results in the split of basket 2 or basket 3. Assume that splitting basket 2 into basket 4 and 5 maximises the difference in the apples in

basket 4 and 5. After splitting basket 2 there are now three baskets that have not yet been split, i.e. basket 3, basket 4 and basket 5. These baskets will be called the "leaf" baskets.

4. A basket may only be split into two new baskets if the number of apples in the new baskets do not fall below a predetermined minimum, which is 20 apples in this case. The process of splitting the best leaf basket with its best splitting question continues until none of the leaf baskets may be split, because splitting will result in two new baskets in which the number of apples is below the predetermined minimum of 20. Imagine that after basket 2 is split the splitting of any leaf baskets will cause the number of apples in the two new baskets to fall below the predetermined minimum of 20. At his point the growing of the decision tree for apples stops and the apples in each of the three leaf baskets are used to create three separate submodels that can be shared by the nine distinct apple models.

To determine which submodel a specific apple model must use, the decision tree for apples is traversed based on the yes/no questions of each basket, until a leaf basket is reached which corresponds to the submodel. For instance, the submodel for normal size yellow apples needs to be determined. The tree is traversed as follows:

1. The colour of this apple model is yellow. The apples in the root noted are split into green apples (basket 2) and non-green apples (basket 3) according to the yes/no question of root basket. Therefore, next basket which might contain the submodel of normal sized yellow apples (if it is a leaf basket) is basket 3.

2. Basket 3 is a leaf node, therefore normal sized yellow apples will use the submodel of basket 3.

To create all the models, a decision tree is grown for each group of models, i.e. apples, pears and bananas. For each distinct apple, pear and banana model, its decision tree is traversed to determine which submodel must be used to create that specific model. This means that different models might share the same submodel.

4.3.2 Description of CART

CART, specifically binary decision trees with splitting questions attached to the nodes, provide an easy method of classifying or partitioning data samples (objects). The classification process is similar to a rule-based system where the classification is carried out by a sequence of decision rules. In the case of determining which state distributions should be tied, the data samples are the specific allophone context of each model. The decision rules are based on the context of the allophones to which the distribution belongs. Decision

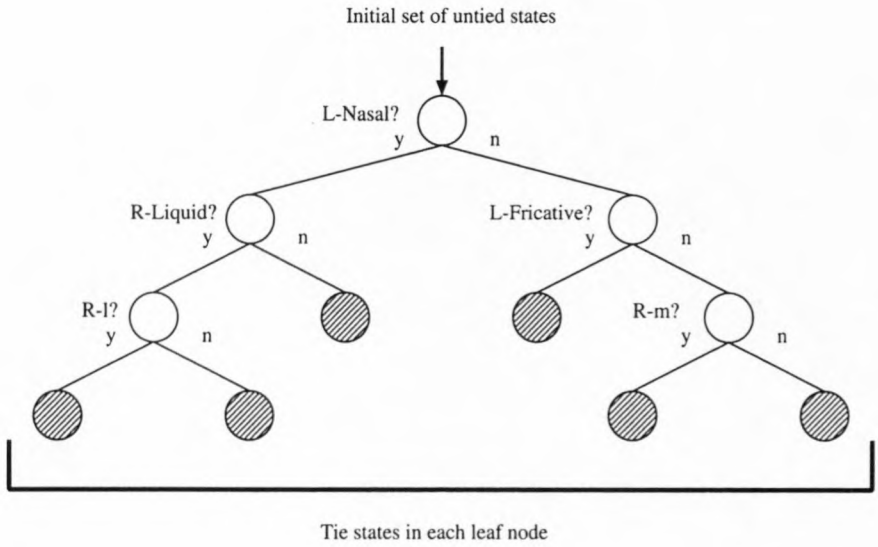

Figure 4.2: *A typical senonic decision tree for the k^{th} HMM states of the set of allophones $\mathbb{C}_\rho$ derived from the base phone ρ.*

tree based state clustering partitions the allophone contexts so that all allophones in a partition shares the same distribution parameters. Illustrated in Fig. 4.2 is the binary decision tree for classifying the state distributions of the k^{th} states of the set of allophones $\mathbb{C}_\rho$ derived from the same base phone ρ. The sequence and number of rules is typically subjectively designed by hand, but CART has the advantage of automatically constructing the decision process based on objective criteria through a data-driven framework. What remains unanswered is exactly how this decision process is constructed or grown?

4.3.3 Growing the CART

When growing a CART (hereafter referred to as a decision tree), the data samples are all initially clustered in the root node (the first leaf node). In the case of decision tree-based state clustering, the data samples $\mathbb{C}_\rho$ represent the set of allophones derived from the base phone ρ. A single sample $\mathcal{C}_\rho$ represents an unique allophone $\mathcal{C}_\rho \in \mathbb{C}_\rho$ and each allophone is modelled with a separate HMM $\Phi_{\mathcal{C}_\rho}$.

A single decision tree $\mathcal{T}_\rho^k$ is constructed for each state k of base phone ρ. The purpose of growing the decision tree is to obtain the set of leaf nodes $\mathbb{L}_\rho^k$. The s^{th} leaf node $\Psi_s^k \in \mathbb{L}_\rho^k$ represents a set of clustered allophones $\mathbb{C}_\rho(s) \subset \mathbb{C}_\rho$ where $s = 1, 2, \ldots, N_{\mathbb{L}}^k$. For each one of the allophones $\mathcal{C}_\rho$ in the cluster $\mathbb{C}_\rho(s)$ there might not be enough training data available to reliably estimate its unshared state distribution $\mathcal{N}_{\text{unshared}}(\mu_{\mathcal{C}_\rho}^k, \Sigma_{\mathcal{C}_\rho}^k)$. Therefore, the data of the allophones in the cluster is pooled in order to estimate the parameters of a single distribution. This *shared* distribution $\mathcal{N}_{\text{shared}}(\mu_s^k, \Sigma_s^k)$ is shared by the k^{th} HMM state of every allophone $\mathcal{C}_\rho$ in the cluster $\mathbb{C}_\rho(s)$. The set of k^{th} HMM states of the allophones in

the cluster $\mathbb{C}_\rho(s)$ is referred to as the set of *tied* HMM states and is denoted by $\mathbb{S}_\rho^k(s)$.

Constructing a globally optimal decision tree is a computationally intractable problem [31]. Therefore, the decision tree is constructed in a locally optimal fashion by splitting or partitioning the data samples clustered in a given leaf node to form two new descendent leaf nodes. When a given leaf node is split during the construction of the decision tree, it is done in order to reduce the overall entropy of the tree. Currently no straightforward entropy measurement for continuous pdfs exist and therefore, the log-likelihood of the training data (given the set of clustered distributions) is used instead (which is explained in more detail in Section 4.2). A given node is split by applying the q^{th} question $\mathbb{Q}(q)$ from a finite set of questions $\mathbb{Q}$ where $q = 1, 2, \ldots N_{\mathbb{Q}}$ and $N_{\mathbb{Q}}$ denotes the number of questions. During any single iteration of the decision tree growing algorithm only a single optimal leaf node Ψ_{s*}^k, whose split with its optimal question $\mathbb{Q}(q^*)$ will result in the largest reduction in entropy, is split.

The optimal leaf node and its optimal splitting question is determined by a currently unspecified splitting criteria $\mathcal{S}$ (which will be defined in Section 4.2). For each of the leaf nodes and for each of the splitting questions, a splitting criterium $\mathcal{S}(s, q)$ is calculated. The leaf node and splitting question which results in the best splitting criterium is selected as the optimal node and question. The decision tree growing process terminates when none of the leaf nodes may be split. A leaf node Ψ_s^k may not be split if one or more of the following conditions are met:

- No more splits are possible because the leaf node's set of clustered allophones $\mathbb{C}_\rho(s)$ consists of only a single allophone.
- The splitting criterium for the split falls below a preset threshold η, i.e.

  $\max_q [\mathcal{S}(s, q)] < \eta$, where $\mathcal{S}(s, q)$ represents the splitting criteria of leaf node Ψ_s^k with question $\mathbb{Q}(q)$ and $q = 1, 2, \ldots N_{\mathbb{Q}}$.
- The state occupancy, $\gamma_\rho^k(s)$ (the measure of the amount of data) in the leaf node Ψ_s^k, falls below a preset threshold τ. This is to assure that the parameters $(\mu_s^k, \boldsymbol{\Sigma}_s^k)$ of the shared distribution represented by leaf node Ψ_s^k can be reliably estimated from the training data.

The tree-growing algorithm is a greedy algorithm because it splits the nodes without regard to any subsequent splits. The decision tree is therefore not constructed in a *globally* optimal fashion, because it is not computationally feasible at this stage. In constructing the tree in a locally optimal fashion it is assumed that the tree will be close enough to being globally optimal.

4.3.4 Question set

In the case of decision tree-based state clustering, $\mathbb{Q}$ is a set of linguistic questions regarding the left and/or right context in which the base phone can occur and therefore, the single question $\mathbb{Q}(q)$ represents a set of allophones. A standard set of questions $\mathbb{Q}$ can be constructed as follows:

1. Each question regards the value of a single variable. Such questions are called *simple* or *singleton* questions.

2. All the questions must be of the form:
   (Is $\mathcal{C}_\rho \in \mathbb{Q}(q)$?) where $\mathbb{Q}(q)$ is any subset of the allophones of all the base phones $\mathbb{C}$.
   The set of allophones $\mathbb{C}_\rho(s)$ clustered in leaf node Ψ_s^k can be split into two new allophone clusters $\mathbb{C}_\rho(\mathcal{D}_1(s,q))$ and $\mathbb{C}_\rho(\mathcal{D}_1(s,q))$ as follows:

$$\begin{aligned} \mathbb{C}_\rho(\mathcal{D}_1(s,q)) &= \{\mathcal{C}_\rho | \mathcal{C}_\rho \in (\mathbb{C}_\rho(s) \cap \mathbb{Q}(q))\} \\ \mathbb{C}_\rho(\mathcal{D}_2(s,q)) &= \mathbb{C}_\rho(s) - \mathbb{C}_\rho(\mathcal{D}_1(s,q)) \end{aligned} \tag{4.1}$$

3. Composite questions can be formed from conjunctions, disjunctions and/or negations of the simple questions.

Complex questions are used to alleviate data fragmentation, but can be implemented by merging leaf nodes that have been formed by using simple questions. Once a set of questions $\mathbb{Q}$ has been constructed it is possible to use domain-specific knowledge to pick a subset of $\mathbb{Q}$ with which to build the decision tree. However, the beauty of CART is the ability to use all possible questions relating to the data being partitioned. This can be done because CART has a statistical data-driven framework that can automatically determine the best questions to use in the decision process.

The data that is being partitioned by the senonic decision tree is the allophone context $\mathcal{C}_\rho$ of each HMM $\Phi_{\mathcal{C}_\rho}$. The questions are chosen on the basis of linguistic knowledge that suggests that certain phones may produce certain types of articulatory effect. Examples of these questions are: *Is the left-context phone a fricative? Is the right-context phone a front vowel?* For each node in the tree, it is checked whether the left or right phone belongs to one of the phonetic categories. The question set used in this research is shown in Appendix B, which was constructed by Odell [31].

4.3.5 Splitting criteria

As mentioned previously, the purpose of decision tree-based state clustering is to share the parameters of continuous distributions. For a given leaf node, the task is normally

to evaluate the entropy reduction of each potential question (split) and to choose the question with the greatest entropy reduction. As a simple entropy measurement for continuous distributions does not exist, a likelihood gain splitting criteria is used. If $\mathbf{X}^{\rho}$ is all the training data for the base phone ρ then the splitting criteria must maximise the total log-likelihood $\mathcal{L}(\mathbb{S}_{\rho}^{k})$ of the training data $\mathbf{X}^{\rho}$, given the set of all tied k^{th} allophone HMM states $\mathbb{S}_{\rho}^{k}$. Performing full maximum likelihood training is computationally very expensive and is therefore not used to determine the likelihoods of the different possible tree architectures. An estimate of the log-likelihood $\mathcal{L}(\mathbb{S}_{\rho}^{k})$ can be found if the following assumptions are made:

- The assignment of training data to states are not altered during the clustering process. The assignment is determined beforehand via the Baum-Welch algorithm.
- The state distributions are assumed to be Gaussian and, for this reason, each set of tied HMM states $\mathbb{S}_{\rho}^{k}(s)$ share a common mean μ_{s}^{k} and covariance $\mathbf{\Sigma}_{s}^{k}$.
- The contribution of transition probabilities to the total log-likelihood is negligible. Although the transition probabilities do have a significant effect on the total likelihood, it will only change the total likelihood if changes occur in the assignment of training data to states.
- The total log-likelihood can be approximated by an average of the log-likelihoods weighted by the probability of state occupancies:

$$\mathcal{L}(\mathbb{S}_{\rho}^{k}) = \sum_{s=1}^{N_{\text{L}}^{k}} \sum_{\mathcal{C}_{\rho} \in \mathbb{C}_{\rho}(s)} \sum_{e=1}^{E^{\mathcal{C}_{\rho}}} \sum_{t=1}^{T^{(e,\mathcal{C}_{\rho})}} \log\left[\mathcal{N}_{\text{shared}}(\mathbf{x}_{t}^{(e,\mathcal{C}_{\rho})}, \mu_{s}^{k}, \mathbf{\Sigma}_{s}^{k})\right] \gamma_{t}^{(e,\mathcal{C}_{\rho})}(k) \tag{4.2}$$

$$\approx \log\left[P(\mathbf{X}^{\rho}, \mathbb{S}_{\rho}^{k})\right] \tag{4.3}$$

where

$E^{\mathcal{C}_{\rho}}$	is the number of training examples of each allophone $\mathcal{C}_{\rho}$,
$\mathbf{X}^{(e,\mathcal{C}_{\rho})}$	is the e^{th} training data example for the allophone $\mathcal{C}_{\rho}$,
$T^{(e,\mathcal{C}_{\rho})}$	is the number of features in the e^{th} training data example $\mathbf{X}^{(e,\mathcal{C}_{\rho})}$

The log-likelihood for a Gaussian distribution is:

$$\begin{aligned}\log\left[\mathcal{N}_{\text{shared}}(\mathbf{x}_{t}^{(e,\mathcal{C}_{\rho})}, \mu_{s}^{k}, \mathbf{\Sigma}_{s}^{k})\right] &= \log\left[(2\pi)^{-\frac{D}{2}} \left|\mathbf{\Sigma}_{s}^{k}\right|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}_{t}^{(e,\mathcal{C}_{\rho})} - \mu_{s}^{k})' \left[\mathbf{\Sigma}_{s}^{k}\right]^{-1} (\mathbf{x}_{t}^{(e,\mathcal{C}_{\rho})} - \mu_{s}^{k})}\right] \\ &= -\frac{1}{2}\Big[D\log(2\pi) + \log(\left|\mathbf{\Sigma}_{s}^{k}\right|) + \\ &\quad (\mathbf{x}_{t}^{(e,\mathcal{C}_{\rho})} - \mu_{s}^{k})' \left[\mathbf{\Sigma}_{s}^{k}\right]^{-1} (\mathbf{x}_{t}^{(e,\mathcal{C}_{\rho})} - \mu_{s}^{k})\Big]\end{aligned} \tag{4.4}$$

Substituting Eq. 4.4 into Eq. 4.2 yields

$$\mathcal{L}(\mathbb{S}_\rho^k) = \sum_{s=1}^{N_\mathrm{L}^k} \sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(s)} \sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^{(e,\mathcal{C}_\rho)}} -\frac{1}{2}\Big[D\log(2\pi) + \log(\left|\boldsymbol{\Sigma}_s^k\right|) + (\mathbf{x}_t^{(e,\mathcal{C}_\rho)} - \mu_s^k)' \left[\boldsymbol{\Sigma}_s^k\right]^{-1} (\mathbf{x}_t^{(e,\mathcal{C}_\rho)} - \mu_s^k)\Big] \gamma_t^{(e,\mathcal{C}_\rho)}(k) \tag{4.5}$$

According to Eqs. 3.33 and 3.35 shared the covariance matrix $\boldsymbol{\Sigma}_s^k$ is re-estimated as

$$\boldsymbol{\Sigma}_s^k = \frac{\sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(s)} \sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^{(e,\mathcal{C}_\rho)}} \gamma_t^{(e,\mathcal{C}_\rho)}(k) \cdot (\mathbf{x}_t^{(e,\mathcal{C}_\rho)} - \mu_s^k)(\mathbf{x}_t^{(e,\mathcal{C}_\rho)} - \mu_s^k)'}{\sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(s)} \sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^{(e,\mathcal{C}_\rho)}} \gamma_t^{(e,\mathcal{C}_\rho)}(k)} \tag{4.6}$$

By cross-multiplying Eq. 4.6 becomes

$$\begin{aligned}
\boldsymbol{\Sigma}_s^k \sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(s)} \sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^{(e,\mathcal{C}_\rho)}} \gamma_t^{(e,\mathcal{C}_\rho)}(k) &= \sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(s)} \sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^{(e,\mathcal{C}_\rho)}} \gamma_t^{(e,\mathcal{C}_\rho)}(k) \cdot (\mathbf{x}_t^{(e,\mathcal{C}_\rho)} - \mu_s^k)(\mathbf{x}_t^{(e,\mathcal{C}_\rho)} - \mu_s^k)' \\
\sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(s)} \sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^{(e,\mathcal{C}_\rho)}} \boldsymbol{\Sigma}_s^k \gamma_t^{(e,\mathcal{C}_\rho)}(k) &= \sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(s)} \sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^{(e,\mathcal{C}_\rho)}} \gamma_t^{(e,\mathcal{C}_\rho)}(k) \cdot (\mathbf{x}_t^{(e,\mathcal{C}_\rho)} - \mu_s^k)(\mathbf{x}_t^{(e,\mathcal{C}_\rho)} - \mu_s^k)' \\
\sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(s)} \sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^{(e,\mathcal{C}_\rho)}} \mathbf{I} \gamma_t^{(e,\mathcal{C}_\rho)}(k) &= \sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(s)} \sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^{(e,\mathcal{C}_\rho)}} \gamma_t^{(e,\mathcal{C}_\rho)}(k) [\boldsymbol{\Sigma}_s^k]^{-1} (\mathbf{x}_t^{(e,\mathcal{C}_\rho)} - \mu_s^k)(\mathbf{x}_t^{(e,\mathcal{C}_\rho)} - \mu_s^k)'
\end{aligned}$$

Taking the trace on both sides and using Theorem 9.1.20 [15, p. 303] which states that:

If $\mathbf{x}$ *is an* $n \times 1$ *vector and* $\mathbf{A}$ *an* $n \times n$ *matrix, then*

$$\mathbf{x}'\mathbf{A}\mathbf{x} = tr(\mathbf{A}\mathbf{x}\mathbf{x}'), \text{ it follows that} \tag{4.7}$$

$$\begin{aligned}
\sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(s)} \sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^{(e,\mathcal{C}_\rho)}} \gamma_t^{(e,\mathcal{C}_\rho)}(k) \mathrm{tr}(\mathbf{I}) &= \sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(s)} \sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^{(e,\mathcal{C}_\rho)}} \mathrm{tr}\left([\boldsymbol{\Sigma}_s^k]^{-1} (\mathbf{x}_t^{(e,\mathcal{C}_\rho)} - \mu_s^k)(\mathbf{x}_t^{(e,\mathcal{C}_\rho)} - \mu_s^k)'\right) \gamma_t^{(e,\mathcal{C}_\rho)}(k) \\
D \sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(s)} \sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^{(e,\mathcal{C}_\rho)}} \gamma_t^{(e,\mathcal{C}_\rho)}(k) &= \sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(s)} \sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^{(e,\mathcal{C}_\rho)}} (\mathbf{x}_t^{(e,\mathcal{C}_\rho)} - \mu_s^k)' \left[\boldsymbol{\Sigma}_s^k\right]^{-1} (\mathbf{x}_t^{(e,\mathcal{C}_\rho)} - \mu_s^k) \gamma_t^{(e,\mathcal{C}_\rho)}(k)
\end{aligned}$$

Therefore, Eq. 4.2 can be simplified to

$$\begin{aligned}
\mathcal{L}(\mathbb{S}_\rho^k) &= \sum_{s=1}^{N_\mathrm{L}^k} -\frac{1}{2}\left[D\log(2\pi) + \log(\left|\boldsymbol{\Sigma}_s^k\right|) + D\right] \sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(s)} \sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^{(e,\mathcal{C}_\rho)}} \gamma_t^{(e,\mathcal{C}_\rho)}(k) \\
&= \sum_{s=1}^{N_\mathrm{L}^k} -\frac{1}{2}\left[D\{1 + \log(2\pi)\} + \log(\left|\boldsymbol{\Sigma}_s^k\right|)\right] \gamma_\rho^k(s) \\
&= \sum_{s=1}^{N_\mathrm{L}^k} \mathcal{L}(\mathbb{S}_\rho^k(s))
\end{aligned} \tag{4.8}$$

where the state occupancy of the leaf node Ψ_s^k is calculated from the training data as:

$$\gamma_\rho^k(s) = \sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(s)} \sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^{(e,\mathcal{C}_\rho)}} \gamma_t^{(e,\mathcal{C}_\rho)}(k) \tag{4.9}$$

Building the decision tree consists of changing the set of shared distributions

$$\mathbb{N}^k_{\rho,\ \text{shared}} = \left\{\mathcal{N}_{\text{shared}}(\mu^k_1, \Sigma^k_1), \ldots, \mathcal{N}_{\text{shared}}(\mu^k_{N^k_{\text{L}}}, \Sigma^k_{N^k_{\text{L}}})\right\} \tag{4.10}$$

to maximise the total likelihood of the training data, whilst ensuring that sufficient data is available to reliably estimate the parameters of each resulting shared distribution. Since splitting a given node (and thus distribution) is assumed to have no effect on the remaining distributions, only the local improvement in the total likelihood need to be calculated. Splitting a leaf node Ψ^k_s with splitting question $\mathbb{Q}(q)$ creates two descendant nodes:

$$\mathbb{D} = \left\{\Psi^k_{\mathcal{D}_1(s,q)}, \Psi^k_{\mathcal{D}_2(s,q)}\right\} \tag{4.11}$$

where $\mathcal{D}_i(s,q)$ indexes parameters of the i^{th} descendent formed by splitting leaf node Ψ^k_s with splitting question $\mathbb{Q}(q)$. The split of the leaf node changes the set of shared distributions $\mathbb{N}^k_{\rho,\ \text{shared}}$ by replacing the parent distribution $\mathcal{N}_{\text{shared}}(\mu^k_s, \Sigma^k_s)$ with two descendant shared distributions, $\mathcal{N}_{\text{shared}}(\mu^k_{\mathcal{D}_1(s,q)}, \Sigma^k_{\mathcal{D}_1(s,q)})$ and $\mathcal{N}_{\text{shared}}(\mu^k_{\mathcal{D}_2(s,q)}, \Sigma^k_{\mathcal{D}_2(s,q)})$. Ignoring for the moment how the state occupancy γ_d and the parameters of the two descendant distributions, μ_d and Σ_d, are determined, the total log-likelihood after the split is calculated as:

$$\begin{aligned}
\mathcal{L}(\{\mathbb{S}^k_\rho - \Psi^k_s\} \cup \mathbb{D}) &= \left[\sum_{i=1,\, i\neq s}^{N^k_{\text{L}}} \mathcal{L}\left(\mathbb{S}^k_\rho(i)\right)\right] + \mathcal{L}(\mathbb{D}) && (4.12)\\
&= -\sum_{i=1,\, i\neq s}^{N^k_{\text{L}}} \frac{1}{2}\left[D\{1+\log(2\pi)\} + \log(\left|\Sigma^k_i\right|)\right]\gamma^k_\rho(i)\\
&\quad - \sum_{d=\mathcal{D}_1(s,q),\mathcal{D}_2(s,q)} \frac{1}{2}\left[D\{1+\log(2\pi)\} + \log(|\Sigma_d|)\right]\gamma_d && (4.13)
\end{aligned}$$

Note that the state occupancy of the parent node Ψ^k_s is equal to the sum of the state occupancies of the descendant nodes i.e.

$$\gamma^k_\rho(s) = \sum_{d=\mathcal{D}_1(s,q),\mathcal{D}_2(s,q)} \gamma_d \tag{4.14}$$

By using Eqs. 4.8, 4.12 and 4.14 the change in the total log-likelihood $\Delta\mathcal{L}(s,q)$ as a result of splitting leaf node Ψ^k_s with question $\mathbb{Q}(q)$, can be calculated as:

$$\begin{aligned}
\Delta\mathcal{L}(s,q) &= \mathcal{L}(\{\mathbb{S}^k_\rho - \Psi^k_s\} \cup \mathbb{D}) - \mathcal{L}(\mathbb{S}^k_\rho)\\
&= \frac{D}{2}\{1+\log(2\pi)\}\left\{\gamma^k_\rho(s) - \sum_{d=\mathcal{D}_1(s,q),\mathcal{D}_2(s,q)} \gamma_d\right\} + \frac{1}{2}\log(\left|\Sigma^k_s\right|)\gamma^k_\rho(s)\\
&\quad - \sum_{d=\mathcal{D}_1(s,q),\mathcal{D}_2(s,q)} \frac{1}{2}\log(|\Sigma_d|)\gamma_d\\
&= \frac{1}{2}\log(\left|\Sigma^k_s\right|)\gamma^k_\rho(s) - \sum_{d=\mathcal{D}_1(s,q),\mathcal{D}_2(s,q)} \frac{1}{2}\log(|\Sigma_d|)\gamma_d && (4.15)
\end{aligned}$$

When Eq. 4.15 is examined, it is clear that the change in the total log-likelihood $\Delta\mathcal{L}_s(q)$, as a result of splitting leaf node Ψ_s^k with question $\mathbb{Q}(q)$, is only the difference between the likelihood of the leaf node Ψ_s^k and its descendant nodes $\mathbb{D}$. Thus, Eq. 4.15 is the likelihood based splitting criteria that is used to determine which question results in the most optimal splitting of a particular node. For this reason the splitting criterium is now specified as:

$$\mathcal{S}(s,q) = \Delta\mathcal{L}(s,q) \tag{4.16}$$

and the optimal leaf node and splitting question is chosen as to maximise the splitting criteria:

$$(s^*, q^*) = \arg\max_{(s,q)} [\Delta\mathcal{L}(s,q)] \tag{4.17}$$

4.3.6 Sufficient statistics for growing the decision tree

The state occupancy and the parameters of the descendant nodes can be calculated directly from the acoustic training data. However, this would require large amounts of storage for all the variables and significant computational overhead. Because the values of γ_d, $\boldsymbol{\Sigma}_d$ and μ_d only depend on the set of unique allophones $\mathbb{C}_\rho(d)$, which are associated with a particular leaf node, they can be calculated from the mean, covariance and state occupancy of each unique allophone $\mathcal{C}_\rho \in \mathbb{C}_\rho(d)$. For the same reason the values of $\gamma_\rho^k(s)$, $\boldsymbol{\Sigma}_s^k$ and μ_s^k are also never computed directly from the acoustic training data. The mean $\mu_{\mathcal{C}_\rho}^k$, covariance $\boldsymbol{\Sigma}_{\mathcal{C}_\rho}^k$ and state occupancy $\gamma_{\mathcal{C}_\rho}^k$, form sufficient statistics to grow a decision tree and are gathered from the acoustic training data *before* the tree growing algorithm starts. The sufficient statistics are calculated using the following MLE equations:

$$\gamma_{\mathcal{C}_\rho}^k = \sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^e} \gamma_t^{(e,\mathcal{C}_\rho)}(k) \tag{4.18}$$

$$\mu_{\mathcal{C}_\rho}^k = \frac{\sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^e} \gamma_t^{(e,\mathcal{C}_\rho)}(k)\mathbf{x}_t^{(e,\mathcal{C}_\rho)}}{\gamma_{\mathcal{C}_\rho}^k} \tag{4.19}$$

$$\Sigma_{\mathcal{C}_\rho}^k = \frac{\sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^e} \gamma_t^{(e,\mathcal{C}_\rho)}(k)(\mathbf{x}_t^{(e,\mathcal{C}_\rho)} - \mu_{\mathcal{C}_\rho}^k)(\mathbf{x}_t^{(e,\mathcal{C}_\rho)} - \mu_{\mathcal{C}_\rho}^k)'}{\gamma_{\mathcal{C}_\rho}^k} \tag{4.20}$$

Using the sufficient statistics of the unique contexts, the state occupancy $\gamma_\rho^k(s)$, mean μ_s^k, and covariance $\boldsymbol{\Sigma}_s^k$ of the shared distribution associated with tied states $\mathbb{S}_\rho^k(s)$, needed in Eq. 4.15, can be respectively calculated as:

$$\gamma_\rho^k(s) = \sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(s)} \gamma_{\mathcal{C}_\rho}^k \tag{4.21}$$

$$\mu_s^k = \frac{\sum_{C_\rho \in \mathbb{C}_s} \mu_{C_\rho}^k \gamma_{C_\rho}^k}{\gamma_s^k} \tag{4.22}$$

$$\Sigma_s^k = \frac{\sum_{C_\rho \in \mathbb{C}_\rho(s)} \left[\Sigma_{C_\rho}^k + (\mu_{C_\rho}^k - \mu_s^k)(\mu_{C_\rho}^k - \mu_s^k)'\right] \gamma_{C_\rho}^k}{\gamma_\rho^k(s)} \tag{4.23}$$

Using sufficient statistics not only has the advantage of faster tree construction, the storage required for all the variables is reduced considerably. Furthermore, the storage required is dependent on the number of unique contexts and is nearly independent of the amount of training data available. Large amounts of training data will only increase the time needed to gather statistics of the unique contexts and also cause larger trees to be grown, because less contexts will need to share distribution parameters.

The state occupancy is an indication of the amount of training data in the node that will be used to estimate the parameters of the tied distribution of that node. The constraint on trainability is thus implemented by assuming that the change in likelihood is zero when γ_d falls below a predetermined threshold. This is implemented in the algorithm by setting $\Delta\mathcal{L}(s, q) = -\infty$.

4.3.7 Construction of models of unseen contexts

After all the decision trees have been constructed, it is relatively easy to construct models of unseen contexts. To construct a model for an unseen allophone context, a transition probability matrix $\mathbf{A}$ and set of state distributions $\mathbf{B}$ are needed. The transition probability matrix is obtained by cloning the matrix A of the base phone ρ from which the unseen allophone context is derived. The state distribution of state k is obtained by traversing the decision tree of state k and base phone ρ, according to the allophone context of the unseen allophone. The distribution associated with the leaf node that will be reached is used for the kth state distribution of the unseen allophone.

4.3.8 Summary of decision tree construction

The algorithms assume that the state distributions are Gaussian distributions. Most modern speech recognition systems uses GMM distributions for the state distributions. The method that is employed to obtain GMM shared state distributions from single shared Gaussians is discussed in Section 5.1.3.5. The algorithm for gathering the sufficient statistics from the acoustic training data in order to grow the decision trees is as follows:

Table 4.1: *The algorithm used to gather sufficient statistics for growing decision trees.*

Inputs:

$\mathbb{C}_\rho$: The set of allophones which occur in the training data and is derived from the base phone[1] ρ

$\Phi_{\mathcal{C}_\rho}$: HMM of allophone $\mathcal{C}_\rho \in \mathbb{C}_\rho$

$E^{\mathcal{C}_\rho}$: Number of training examples of allophone $\mathcal{C}_\rho$

$\mathbf{X}^{(e,\mathcal{C}_\rho)}$: Training data examples of allophone $\mathcal{C}_\rho$, $e = 1, 2, \ldots, E^{\mathcal{C}_\rho}$

$\mathcal{T}^{(e,\mathcal{C}_\rho)}$: Complete transcriptions[2] of the acoustic data $\mathbf{X}^{(e,\mathcal{C}_\rho)}$, $e = 1, 2, \ldots, E^{\mathcal{C}_\rho}$

Known parameters:

$\mu^k_{\mathcal{C}_\rho}$: Mean of the distribution of state k of HMM $\Phi_{\mathcal{C}_\rho}$

$\Sigma^k_{\mathcal{C}_\rho}$: Covariance of the distribution of state k of HMM $\Phi_{\mathcal{C}_\rho}$

$T^{(e,\mathcal{C}_\rho)}$: Length of the e^{th} example, $\mathbf{X}^{(e,\mathcal{C}_\rho)}$, of allophone $\mathcal{C}_\rho$

Computation of sufficient statistics: $\forall\, \mathcal{C}_\rho,\ \mathcal{C}_\rho \in \mathbb{C}_\rho$

Compute $\gamma_t^{(e,\mathcal{C}_\rho)}(k)$ for $e = 1, 2, \ldots, E^{\mathcal{C}_\rho}$ using the Baum-Welch algorithm

Compute the state occupancy $\gamma^k_{\mathcal{C}_\rho}$ of state k of allophone $\mathcal{C}_\rho$

$$\gamma^k_{\mathcal{C}_\rho} = \sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^{(e,\mathcal{C}_\rho)}} \gamma_t^{(e,\mathcal{C}_\rho)}(k)$$

Compute the mean $\mu^k_{\mathcal{C}_\rho}$ of state k of allophone $\mathcal{C}_\rho$

$$\mu^k_{\mathcal{C}_\rho} = \left[\sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^{(e,\mathcal{C}_\rho)}} \gamma_t^{(e,\mathcal{C}_\rho)}(k) \mathbf{x}_t^{(e,\mathcal{C}_\rho)} \right] / \gamma^k_{\mathcal{C}_\rho}$$

Compute the covariance $\mathbf{\Sigma}^k_{\mathcal{C}_\rho}$ of state k of allophone $\mathcal{C}_\rho$

$$\mathbf{\Sigma}^k_{\mathcal{C}_\rho} = \left[\sum_{e=1}^{E^{\mathcal{C}_\rho}} \sum_{t=1}^{T^{(e,\mathcal{C}_\rho)}} \gamma_t^{(e,\mathcal{C}_\rho)}(k) (\mathbf{x}_t^{(e,\mathcal{C}_\rho)} - \mu^k_{\mathcal{C}_\rho})(\mathbf{x}_t^{(e,\mathcal{C}_\rho)} - \mu^k_{\mathcal{C}_\rho})' \right] / \gamma^k_{\mathcal{C}_\rho}$$

The algorithm in Table 4.1 computes the sufficient statistics for the data for a large number of HMMs. Specifically one HMM for each allophone $\mathcal{C}_\rho$ in the training data. The algorithm for growing a decision tree for state k of base phone ρ from the sufficient statistics gathered from the training data, is as follows:

[1]The algorithm is also applicable to allophones derived from base phonemes. The choice of phones vs. phonemes depends on whether the database was phonetically or phonemically transcribed.

[2]If complete transcriptions are not available, they can be created by using forced alignment as discussed in Section 5.1.1.2.

Table 4.2: *The algorithm for growing a decision tree for state k of base phone ρ.*

Inputs:

$\gamma^k_{\mathcal{C}_\rho}$	: State occupancy on state k of allophone HMM $\Phi_{\mathcal{C}_\rho}$
$\mu^k_{\mathcal{C}_\rho}$	: Mean of distribution on state k of allophone HMM $\Phi_{\mathcal{C}_\rho}$
$\Sigma^k_{\mathcal{C}_\rho}$	: Covariance of distribution on state k of allophone HMM $\Phi_{\mathcal{C}_\rho}$
$\mathbb{Q}$	: The set of linguistic questions regarding the left and/or right context of the base phones. $\mathbb{Q}(q) \in \mathbb{Q}$ denotes the q^{th} question. $\mathbb{Q}(q)$ consists of a set of allophones.
$N_{\mathbb{Q}}$	: The number of linguistic questions
$\mathcal{C}_\rho$	: Allophone $\mathcal{C}_\rho \in \mathbb{C}_\rho$ derived from base phone ρ
N_ρ	: The number of allophones derived from ρ
τ	: The minimum node occupancy threshold

Initialisation:

$\mathbb{C}_\rho(0) = \mathbb{C}_\rho$	: All allophones derived from base phone ρ are clustered in the first leaf node Ψ^k_0 called the root node. Ψ^k_s denotes the s^{th} leaf node. $\mathbb{C}_\rho(s)$ denotes the set of allophones clustered in the leaf node Ψ^k_s.
$\mathbb{L}^k_\rho = \{\Psi^k_0\}$	: The set of leaf nodes for the first iteration contains only the root node.
$N^k_{\mathbb{L}} = 1$	: The number of leaf nodes for the 1st iteration is set to one. $N^k_{\mathbb{L}}$ denotes the number of leaf nodes during each iteration.

Repetition:

for $s = 1, 2, \ldots, N^k_{\mathbb{L}}$: Iterate through all current leaf nodes

(a) Compute the parameters of the leaf node Ψ^k_s

$$\gamma^k_s = \sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(s)} \gamma^k_{\mathcal{C}_\rho}$$

$$\mu^k_s = \left[\sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(s)} \mu^k_{\mathcal{C}_\rho} \gamma^k_{\mathcal{C}_\rho} \right] / \gamma^k_s$$

$$\mathbf{\Sigma}^k_s = \left[\sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(s)} \{\mathbf{\Sigma}^k_{\mathcal{C}_\rho} + (\mu^k_{\mathcal{C}_\rho} - \mu^k_s)(\mu^k_{\mathcal{C}_\rho} - \mu^k_s)'\} \gamma^k_{\mathcal{C}_\rho} \right] / \gamma^k_s$$

(b) for $q = 1, 2, \ldots, N_{\mathbb{Q}}$: Iterate through all questions

(i) Split the allophone cluster $\mathbb{C}_\rho(s)$ of leaf node Ψ_s^k with question $\mathbb{Q}(q)$ to create two new allophone clusters $\mathbb{C}_\rho(\mathcal{D}_1(s,q))$ and $\mathbb{C}_\rho(\mathcal{D}_2(s,q))$ where $\mathcal{D}_i(s,q)$ is an index to the i^{th} new allophone cluster.

$$\mathbb{C}_\rho(\mathcal{D}_1(s,q)) = \{\mathcal{C}_\rho | \mathcal{C}_\rho \in (\mathbb{C}_\rho(s) \cap \mathbb{Q}(q))\}$$
$$\mathbb{C}_\rho(\mathcal{D}_2(s,q)) = \mathbb{C}_\rho(s) - \mathbb{C}_\rho(\mathcal{D}_1(s,q))$$

(ii) Compute the parameters of the new allophone clusters of leaf node Ψ_s^k as a result of the split with question $\mathbb{Q}(q)$.

for $d = \mathcal{D}_1(s,q), \mathcal{D}_2(s,q)$

$$\gamma_d^k = \sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(d)} \gamma_{\mathcal{C}_\rho}^k$$

$$\mu_d^k = \left[\sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(d)} \mu_{\mathcal{C}_\rho}^k \gamma_{\mathcal{C}_\rho}^k \right] / \gamma_d^k$$

$$\boldsymbol{\Sigma}_d^k = \left[\sum_{\mathcal{C}_\rho \in \mathbb{C}_\rho(d)} \{\boldsymbol{\Sigma}_{\mathcal{C}_\rho}^k + (\mu_{\mathcal{C}_\rho}^k - \mu_d^k)(\mu_{\mathcal{C}_\rho}^k - \mu_d^k)'\} \gamma_{\mathcal{C}_\rho}^k \right] / \gamma_d^k$$

(iii) Compute the log-likelihood increase of splitting node Ψ_s^k with question $\mathbb{Q}(q)$.

$$\begin{cases} \Delta\mathcal{L}(s,q) = \frac{1}{2}\log(|\boldsymbol{\Sigma}_s^k|)\gamma_s^k - \sum_{d=\mathcal{D}_1(s,q),\mathcal{D}_2(s,q)} \frac{1}{2}\log(|\boldsymbol{\Sigma}_d^k|)\gamma_d^k & \forall\, \gamma_d^k \geq \tau \\ \Delta\mathcal{L}(s,q) = -\infty, & \exists\, \gamma_d < \tau \end{cases}$$

$(s^*, q^*) = \arg\max_{(s,q)} [\Delta\mathcal{L}(s,q)]$: Determine the best leaf node $\Psi_{s^*}^k$ to split with its best splitting question $\mathbb{Q}(q^*)$

$\mathbb{D}^* = \left\{ \Psi_{\mathcal{D}_1(s^*,q^*)}^k, \Psi_{\mathcal{D}_2(s^*,q^*)}^k \right\}$: The descendants of the leaf node $\Psi_{s^*}^k$ where $\mathcal{D}_i(s^*, q^*)$ denotes the two new leaf node indexes.

$\mathbb{L}_\rho^k = \left\{ (\mathbb{L}_\rho^k - \Psi_{s^*}^k) \cup \mathbb{D}^* \right\}$: Replace the parent leaf node $\Psi_{s^*}^k$ with its descendants to form the new set of leaf nodes. The parent leaf node $\Psi_{f^*}^k$ becomes an internal node of the tree.

$N_{\mathbb{L}}^k = N_{\mathbb{L}}^k + 1$: Increase the number of leaf nodes by one

Termination:

$\Delta\mathcal{L}(s^*, q^*) = -\infty$: No more leaf nodes can be split

$\mathbb{L}_\rho^k$: The mean and covariance of each leaf node is the parameters of each shared Gaussian distribution.

4.4 Summary

In this chapter the motivations behind modelling context in speech were discussed. This chapter started by stating the selection criteria for context-dependent models and continues with a discussion of the different types of context-dependent units. The problems associated with context-dependent modelling were discussed next, focusing particularly on the problem of obtaining reliable parameters from the training data and the problem of handling context-dependent models that are not found in the training data. For both of these problems, it is argued that parameter sharing is a viable solution.

Based on the selection criteria, phonemes are chosen as context-dependent units. It is argued that decision tree-based state clustering is a suitable technique for implementing parameter sharing based on its advantages when handling the problems associated context-dependent models. The rest of the chapter was dedicated to the theory and implementation of decision tree-based state clustering.

Chapter 5

Implementation

Before isolated or continuous speech recognition can be performed, it is necessary to construct the speech models to which the unknown speech are compared. In this chapter it is explained how the theory and algorithms of the previous three chapters are used to model English phonemes.

5.1 Construction of phonetic-acoustic models

In order to construct the acoustic models it is necessary to have access to some form of training data. The training data need to be representative of the speech that is being modelled. Two types of training methods can be distinguished: *supervised* and *unsupervised* training. Denote the pair $(\mathbf{X}, w_i)$ as a sample, where $\mathbf{X}$ is the observed data and w_i is the class (word or phoneme) to which the data $\mathbf{X}$ belongs. During *supervised* training the information w_i about the class of the observed data sample $\mathbf{X}$ is given, in contrast with *unsupervised* training where this information is not available. Sample data in which the class information w_i is available is referred to as labelled or complete data. Supervised training algorithms include Maximum Likelihood Estimation (MLE), while unsupervised training algorithms include the Vector Quantisation (VQ), the Expectation Maximisation (EM) algorithm and multivariate Gaussian mixture distribution estimation. It will be necessary to use both supervised and unsupervised training when constructing the acoustic models.

For the speech community, transcription of a spoken corpus is linked to the notion of labelling. Therefore, the labels of the speech data are sometimes referred to as transcriptions. For the speech community, the transcription of a speech signal denotes the temporal definition and labelling of its parts with reference to the acoustic signal. These "parts" may be temporarily discrete or overlapping and may be defined in acoustic, physiological, phonetic or higher-level linguistic terms (such as words). When a transcription contains only labelling information and not temporal information (i.e. the sequence of labels is

known, but not the start and end times of the individual labels), the transcription will be referred to as a non-time-aligned or incomplete transcription. When a transcription contains both temporal and labelling information, the transcription will be referred to as a time-aligned or complete transcription.

In this thesis, the following definitions of different types of transcriptions are used:

- **Orthographic**: in which the standard spelling of a given word is used to indicate the spoken words.
- **Phonemic**: in which phoneme strings are derived from the orthographic transcription, using some mapping of word to phoneme strings.
- **Allophonic**: in which different symbols are used for a single phoneme when this phoneme occurs in different contexts.
- **Phonetic**: which attempts to represent the string of phonemes which the speaker actually spoke.

For example, the incomplete orthographic transcription of the utterance "fifty five years" is:

fifty five years

The incomplete phonemic transcription of the same utterance is:

/f ə f t ɪ/ [sil]? /f ʌ͡ɪ v/ [sil]? /j ɜː z/

where the optional silence is denoted by the label "[sil]?". A complete transcription of the speech signal contains the beginning and end times of each transcription label. Because the end time of one label invariably coincides with the beginning time of the next, it is only necessary to specify the end times of each transcription label. Using the previous example, the complete phonemic transcription of the utterance "fifty five years" is:

0.36275 /f/
0.47168 /ə/
0.58061 /f/
0.68954 /t/
0.79847 /ɪ/
0.90740 [sil]
1.01633 /f/
1.12527 /ʌ͡ɪ/
1.23420 /v/
1.34313 [sil]
1.45206 /j/
1.56099 /ɜː/
1.66992 /z/

It would be preferable to use a speech corpus in which all the transcriptions are complete phonetic or phonemic transcriptions, but the cost of creating such a speech corpus is prohibitive. Therefore, there are speech corpora in existence for which only a part of the speech corpus' transcriptions are complete. The rest of the speech corpus' transcriptions are incomplete. For this reason the method used to construct the acoustic models must be able to use transcriptions which are incomplete.

The transcriptions of the South African English speech corpus are incomplete phonemic transcriptions. For more information on the speech corpora refer to Appendix C.

At the time this research was conducted, there was no complete phonetic transcriptions available for the South African English corpus. Therefore, it was decided to use the NTIMIT speech corpus to initialise the acoustic models. The NTIMIT speech corpus contains complete orthographic and phonetic transcriptions, which are labelled using the ARPABET phonetic alphabet. The initialisation of the acoustic models is done by mapping a previously created set of NTIMIT phoneme models to the new alphabet, which is a one-to-many mapping. These mapped phoneme models are used to create complete phonemic transcriptions of the South African English corpus, using the signal processing techniques that have been trained on the NTIMIT speech corpus. New acoustic models are subsequently created using only the complete phonemic transcriptions of the South African English corpus.

5.1.1 Automatic alignment of incomplete transcriptions

Thirty-nine context-independent phonemes are created from the NTIMIT speech corpus using the method discussed in the next section. In the case of South African English the

discrepancy between the American and South African dialect should not have too large an influence on the South African English models, because the NTIMIT phonemes are only used to automatically align the incomplete transcriptions. Although the phonemes based on the American dialect are used to align the transcriptions, there is no direct influence of the American dialect on the South African dialect since the acoustic models will be trained solely on the South African speech data.

5.1.1.1 Mapping of phonemes

The 39 NTIMIT phonemes are mapped to the desired phoneme set using the following mapping:

Table 5.1: *NTIMIT to IPA phoneme mapping.*

Desired set	NTIMIT Initialisation		Desired set	NTIMIT Initialisation	
IPA	ARPABET	IPA	IPA	ARPABET	IPA
/a/	ao	/ɔ/	/ɒ/	ao	/ɔ/
/æ/	ae	/æ/	/a͡ʊ/	aw	/a͡ʊ/
/a͡i/	ay	/a͡ɪ/	/a͡u/	oy	/ɔ͡ɪ/
/aː	ao	/ɔ/	/ɑ/	ao	/ɔ/
/ɑː/	ao	/ɔ/	/b/	b	/b/
/ɓ/	b	/b/	/b̥/	b	/b/
/cʰ/	t	/t/	/c'/	t	/t/
/ǀʰ/	t	/t/	/ǀ̃/	t	/t/
/ǀ̃̆/	t	/t/	/ǀ/	t	/t/
/ǀ̆/	t	/t/	/ǃʰ/	k	/k/
/ǃ/	k	/k/	/ǃ̃/	k	/k/
/ǃ̃̆/	k	/k/	/ǃ̆/	k	/k/
/ǁʰ/	k	/k/	/ǁ/	k	/k/
/ǁ̃/	k	/k/	/ǁ̃̆/	k	/k/
/ǁ̆/	k	/k/	/c/	t	/t/
/ɔ/	ao	/ɔ/	/ɔ͡ɪ/	oy	/ɔ͡ɪ/
/ɔː/	ao	/ɔ/	/d/	d	/d/
/ð/	dh	/ð/	/d͡z/	jh	/ʤ/
/d͡ʒ/	jh	/ʤ/	/d͡z/	jh	/ʤ/
/d̥/	d	/d/	/e/	eh	/ɛ/
/e͡ɪ/	ey	/e͡ɪ/	/ɛ/	eh	/ɛ/
/ɛː/	eh	/ɛ/	/ɜː/	ih	/i/
/f/	f	/f/	/g/	g	/g/
/ʔ/	t	/t/	/h/	hh	/h/

Desired set	NTIMIT Initialisation	
IPA	ARPABET	IPA
/ɦ/	hh	/h/
/ɪ/	ih	/ɪ/
/i/	iy	/i/
/iː/	iy	/i/
/j/	y	/j/
/k^h/	k	/k/
/k/	k	/k/
/ɬ/	sh	/ʃ/
/ɮ/	sh	/ʃ/
/m̩/	m	/m/
/n̩/	n	/n/
/o/	ow	/o/
/p'/	p	/p/
/r/	r	/r/
/ʃ/	sh	/ʃ/
/ə/	ah	/ʌ/
/t^h/	t	/t/
/θ/	th	/θ/
/t͡s'/	s	/s/
/t͡ʃ/	ch	/ʧ/
/u͡a/	uw	/u/
/u/	uw	/uw/
/ɒ͡ɪ/	ah	/ʌ/
/w/	w	/w/
/ʒ/	sh	/ʃ/

Desired set	NTIMIT Initialisation	
IPA	ARPABET	IPA
/ʊ/	uh	/ʊ/
/ɪ͡ə/	ih	/ɪ/
/i͡a/	iy	/i/
/ɟ/	d	/d/
/j/	y	/j/
/k'/	k	/k/
/k͡x'/	k	/k/
/l/	l	/l/
/m/	m	/m/
/ŋ/	ng	/ŋ/
/n/	n	/n/
/p^h/	p	/p/
/p/	p	/p/
/ɹ/	r	/r/
/s/	s	/s/
/ə͡ʊ/	ow	/o/
/t'/	t	/t/
/t͡ɬ'/	sh	/ʃ/
/t͡ʃ^h/	ch	/ʧ/
/t/	t	/t/
/uː/	uw	/u/
/ɒ/	ah	/ʌ/
/v/	v	/v/
/x/	sh	/ʃ/
/z/	z	/z/

These mapped phoneme models are used to determine the optimal boundaries between individual labels in the incomplete phonemic transcriptions. This is demonstrated in the next section.

5.1.1.2 Forced alignment

As previously mentioned, the South African English transcriptions are incomplete phonemic transcriptions, but the non-linguistic events are marked in the transcriptions. The non-linguistic events include noise sources such as silence, speaker noise, intermittent noise and stationary noise. For a more complete description of the non-linguistic events refer to Appendix C. Optional non-linguistic events between words are also indicated.

The purpose of forced alignment is to automatically estimate time alignments for the individual labels of the incomplete phonemic transcriptions. Let a single utterance, for example, correspond to the sequence of phonemes $p_1 p_2 p_3 \ldots p_N$. A utterance HMM model is created by concatenating the HMM models corresponding to the phoneme sequence as is illustrated in Fig. 5.1. The optimal path through the utterance model is computed by

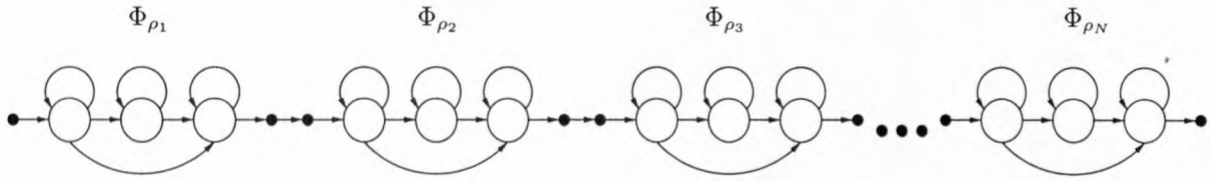

Figure 5.1: *An utterance HMM.*

using the *Viterbi* algorithm. By keeping track of the feature vector-to-state assignments and therefore the feature vectors-to-phoneme model assignments, the speech-to-phoneme time alignments can be extracted from the optimal path. The incomplete phonemic transcriptions are automatically time-aligned by using the NTIMIT based phoneme models. This is done by forcing the speech utterance to correspond to the optimal path through the utterance model.

The resolution of the time alignment is restricted by the length of the analysis frame used when extracting features from the raw speech data, as well as the length by which the frame is shifted. However, if the frame length becomes too small the frequency characteristics of the speech will be smeared across time. A higher time alignment resolution can be achieved by shortening the length by which each analysis frame is shifted. However, this would result in a larger number of feature vectors and therefore the recognition time would increase.

5.1.2 Construction of context-independent phonemes

The context-independent acoustic model construction method assumes that complete phoneme transcriptions of the speech data are available. This is the case for the NTIMIT speech corpus, but not for the South African English corpus. Before the South African English context-independent acoustic models are constructed, the incomplete phonemic transcriptions of the South African English corpus are automatically aligned by using the forced alignment technique. The context-independent acoustic model construction method followed in this thesis is outlined as follows:

1. **Initialisation**: for each of the acoustic models, collect its observed data from the complete phonemically transcribed data and use the re-estimation algorithms (discussed in Section 3.3) to estimate the model's parameters.

2. **Embedded re-estimation**: for each utterance of transcribed speech, concatenate the initialised acoustic models to form a sentence HMM based on the incomplete phonemic transcription of the utterance. Use the re-estimation algorithms to automatically determine the probabilistic alignment of phonemes. Re-estimate the models' parameters based on the state-to-speech alignments.

5.1.2.1 Initialisation

Two types of parameters need to be initialised before the phoneme HMMs can be trained. The first type of parameter is the probability distributions and the second is the state transitions probabilities. Once the parameters are initialised new model parameters can be estimated using the *Viterbi* or *Baum-Welch* EM algorithms.

The pdfs are initialised from complete phonemic transcriptions that have been subdivided. For instance, if a phoneme model consists of a three-state, left-to-right HMM, each occurance of that phoneme in the transcriptions will be equally divided into three segments. In this research both single and mixture Gaussian distributions are used as state output probability distribution functions (pdfs). The single Gaussian distribution is initialised by simply collecting all data corresponding to that pdf and then using MLE to estimate a mean and covariance matrix. An M-mixture Gaussian distribution is initialised by collecting all data corresponding to that pdf and then performing binary split vector quantisation to divide the data into M groups. From these M groups a vector codebook is derived, which corresponds to the M means of the data groups. These M means are used as an initial estimate of the means of the M single Gaussian distributions of which the mixture Gaussian distribution consists. The covariance matrix of each Gaussian distribution is set to unity. Once an initial mixture Gaussian distribution has been created for each state of each phoneme, the pdfs are trained iteratively by collecting all data corresponding to the specific pdfs and using the EM algorithm

After proper initial estimates of the pdf parameters have been obtained, HMM models Φ_p are constructed for each phoneme p. The initial choice of transition probabilities is less important than having a proper initial estimate of the pdf parameters. The transition probabilities of a state are chosen in such a way that the state self-loop probability is the highest. The self-loop probability is initialised to

$$a_{ii} = 0.8 \tag{5.1}$$

while the remaining transition probabilities are initialised to

$$a_{ij} = \frac{0.2}{\text{number of links exiting state}}, \quad i \neq j \tag{5.2}$$

The phoneme HMMs are trained directly from the complete phonemic transcriptions by collecting all occurances of each phoneme and using the re-estimation algorithms on each

phoneme HMM separately. The number of training iterations carried out is determined by the minimum total improvement in models, or a predetermined maximum number of iterations. The total improvement in the models is the difference in total log-likelihood between the current and previous iteration. The total log-likelihood of an iteration is computed by summing the log-likelihood of all the occurances given the models, i.e.

$$\mathcal{L}_n = \log\left[\prod_{p=1}^{P}\prod_{e=1}^{E^p} P(\mathbf{X}_p^e|\Phi_p)\right] = \sum_{p=1}^{P}\sum_{e=1}^{E^p} \log P(\mathbf{X}_p^e|\Phi_p) \tag{5.3}$$

where $\mathbf{X}_p^e$ is the e^{th} example of the p^{th} phoneme. The total improvement is thus

$$\Delta\mathcal{L} = \left|\frac{\mathcal{L}_n - \mathcal{L}_{n-1}}{\mathcal{L}_{n-1}}\right| \tag{5.4}$$

If the total improvement falls below a given threshold (0.0005 in this case), the training is halted even before the maximum number of iterations has been performed.

5.1.2.2 Embedded re-estimation

The complete phonemic transcriptions are created by forcibly aligning the time boundaries based on the NTIMIT mapped phoneme models. By initialising and training on this forcibly aligned transcriptions the implicit assumption is that the time boundaries are correct. Even if the complete phonemic transcriptions are transcribed by linguists, it is quite possible that the time boundaries would not be perfectly correct. In order to compensate for the possible errors in time boundaries, it is necessary to perform *embedded* re-estimation. Embedded re-estimation is very similar to forced alignment in the sense that for each training utterance an utterance HMM is created from a bank of phoneme models. For example, let the training data consist of 10 utterances. One iteration of embedded training consists of the following:

1. For each of the utterances U^e construct an utterance HMM Φ^e by concatenating the phoneme HMMs corresponding to the phoneme sequence of the utterance. The phoneme models are all taken from a bank of phoneme models. The same phoneme, in different positions, in different utterances, corresponds to the same phoneme model in the model bank.

2. Determine state occupancy $\gamma_t^e(i)$ of the e^{th} utterance by using the Baum-Welch or Viterbi algorithm. The Viterbi algorithm assigns the feature vector at time t to a single state, while the Baum-Welch algorithm assigns the feature vector at time t, $\mathbf{X}_t^e$ and a weight $\gamma_t^e(i)$ to all the states.

3. By keeping track of the feature-to-state assignments, collect all the data corresponding to each pdf and state transition. Data assigned to the same phoneme in different positions, in different utterances, are assigned to the same phoneme model.

4. Estimate new model parameters by using the re-estimation formulas (Section 3.3).

The number of training iterations carried out is determined by the minimum total improvement in models, or a predetermined maximum number of iterations. The total improvement in the models is the difference in total log-likelihood between the current and previous iteration. The total log-likelihood of an iteration is computed by summing the log-likelihood of all the utterances given the models i.e.

$$\mathcal{L}_n = \log \left[\prod_{e=1}^{E} P(U^e|\Phi^e) \right] = \sum_{e=1}^{E} \log P(U^e|\Phi^e) \tag{5.5}$$

and the total improvement is thus

$$\Delta\mathcal{L} = \left| \frac{\mathcal{L}_n - \mathcal{L}_{n-1}}{\mathcal{L}_{n-1}} \right| \tag{5.6}$$

If the total improvement falls below a given threshold (0.0005 in this case), the training is halted even before the maximum number of iterations has been performed.

5.1.3 Construction of context-dependent phonemes

The context-dependent acoustic model construction method followed in this thesis is outlined as follows:

1. **Initialisation**: create each of the context-dependent acoustic models by cloning its base context-independent model. Collect each of the acoustic models' observed data from the complete allophonically-transcribed data and use the re-estimation algorithms (discussed in Section 3.3) to estimate the model parameters. If there is not enough data to reliably estimate a certain model's parameters, do not update that model. The complete allophonic transcriptions are easily derived from the complete phonemic transcriptions.

2. **Gathering of statistics**: for each context-dependent acoustic model, gather the sufficient statistics (i.e. occupancy, mean and covariance of each state) needed to construct decision trees from the complete phonemically transcribed data.

3. **Tree construction and parameter sharing**: for each state of each base context-independent model, construct a binary decision tree using the techniques of Chapter 4. Share the model parameters of context-dependent acoustic models, derived from the same base context-independent acoustic model, by using the constructed binary decision trees.

4. **Embedded re-estimation**: for each utterance of transcribed speech, concatenate the shared acoustic models to form a sentence HMM based on the incomplete

phonemic transcription of the utterance. Use the re-estimation algorithms to automatically determine the segmentation. Re-estimate the models' parameters based on the state-to-speech alignments.

5. **Mixture component splitting**: change the state output pdfs to $\mathcal{M}$-mixture GMMs by using the Viterbi algorithm to collect the state-to-speech alignment and by vector quantisating the corresponding data to obtain the initial means for the GMMs. Re-estimate the GMMs' parameters by using the normal re-estimation algorithms and embedded re-estimation.

5.1.3.1 Initialisation

Each base monophone represents a base class from which context-dependent phonemes can be derived. For each of the base classes, the distinct contexts in both the training data and the complete database are identified. HMMs are created for each distinct context by cloning the base monophone. For example, if the base monophone /ə/ has 200 distinct contexts, 200 copies of the /ə/ monophone HMM are made. In this state none of the context-dependent HMMs share any model parameters. Referring back to Section 4.3, the parameter sharing technique, decision tree-based state clustering, derives its decision trees from HMMs that use full-covariance Gaussian distributions as state output pdfs. Therefore, the context-dependent HMMs can only be cloned from context-independent HMMs that meet the requirement of having full-covariance Gaussian state output pdfs. All context-dependent HMMs derived from the same base class have the same topology.

As is the case with the initialisation of the context-independent HMMs, the context-dependent phoneme HMMs are initialised by training directly on the complete allophonic transcriptions. This is done by collecting all occurances of each phoneme and using the re-estimation algorithms on each phoneme HMM separately. Because of the scarcity of data, some of the context-dependent HMMs will not have enough training data to reliably estimate new model parameters. If a certain HMM does not have enough data, the model parameters are left unchanged.

5.1.3.2 Gathering of statistics

In order to construct the binary decision trees, it is necessary to gather certain statistics from the training data. The binary decision trees are needed to determine the sharing of model parameters between context-dependent phonemes derived from the same base monophone (refers back to Chapter 4). The statistics that are needed are the mean, covariance and state occupancy of each HMM state of each unique phoneme context. The threshold τ determines the minimum state occupancy of the leaf nodes of the decision trees and therefore determines the level of sharing between models. Different levels of sharing

result in different recognition performances and therefore it is necessary to determine the optimal threshold τ^* by performing multiple phoneme recognition experiments. It is, however, only necessary to gather and store the statistics once, as decision trees with different state occupancy thresholds τ can be grown from the same set of statistics.

The statistics are gathered by using the Baum-Welch algorithm. From the complete allophonic transcriptions, the T feature vectors $\mathbf{X}_1^T = \mathbf{X}^{(e,\mathcal{C}_\rho)}$ corresponding to the e^{th} example of each unique allophone context $\mathcal{C}_\rho$ in the training data, are collected. For each example of each unique allophone context in the training data the following is done:

1. Given the context-dependent HMM $\Phi_{\mathcal{C}_\rho}$ calculate the likelihood $f(\mathbf{X}_1^T|\Phi_{\mathcal{C}_\rho})$ and state occupancy $\gamma_t(i)$ using the Baum-Welch algorithm.

2. Assign the feature vector $\mathbf{X}_t$ at time t and a weight $\gamma_t(i)$ to state i.

When all the training data has been processed, the mean, covariance and state occupancy of each unique context is calculated using MLE Eqs. 4.18, 4.19 and 4.20.

When gathering the statistics, the feature vectors are not collected in the computer's memory as this would require too much storage space. Instead, the first- and second-order moments (i.e. $E[\mathbf{X}]$ and $E[\mathbf{X}^2]$), which are the sufficient statistics for the MLE of the mean and covariance, are accumulated as each example is processed. The data scarcity problem will cause some contexts to have insufficient data, causing ill-conditioned covariance matrices and low state occupancies. Fortunately, the effect of the poorly estimated model parameters will not influence the shared mean and covariance of the decision tree leaf nodes, as the mean and covariance of a context is weighed by its state occupancy (Eqs. 4.22 and 4.23). However, practically it was found that the determinant of the ill-conditioned covariance matrices become negative. In order to avoid negative determinants it is necessary to guarantee that the estimated covariance matrices are always positive definite. Therefore, the two-pass method of accumulating the sufficient statistics for the MLE of covariance matrices is used.

5.1.3.3 Tree construction and parameter sharing

Using the sufficient statistics gathered in the previous step, binary decision trees are constructed for each state of each base class. The decision trees of each base class are constructed independently from the other base classes, which reduces the amount of storage needed during construction of the decision trees. If, for example, the HMM topology of the base class has three states, three decision trees are constructed (one for each state). It is important to realise that only the unique allophone contexts that occur in the training data are used in the construction of the decision trees. When the decision trees of a base class have been constructed, the context-dependent HMMs are reconfigured to share state pdf parameters. As mentioned in Section 4.3.7, for the specific state of a

context-dependent HMM, the shared state output pdf is determined by traversing the decision tree corresponding with that state until a leaf node is reached. For example, imagine the decision tree of the first state of the /ə/ base class has only 20 leaf nodes. The state output pdf of the first state of all 200 unique /ə/ contexts must be one of the 20 shared pdfs corresponding to the 20 leaf nodes of the decision tree. Although multiple context-dependent HMMs refer to the same pdf, only 20 pdfs exist for the first state of the /ə/ context-dependent HMMs. The unseen contexts, which occur in the complete data but not in the training data, are constructed as well. Only the state pdf parameters are shared and each context-dependent HMM has its own state transition probability matrix.

5.1.3.4 Embedded re-estimation

As with the construction of the context-independent HMMs, it is necessary to perform *embedded* re-estimation to compensate for possible errors in the time boundaries of the complete allophonic transcriptions. The same methodology is used, except that the allophonic transcriptions need to be derived from the phonemic transcriptions so that the transcriptions indicate context dependency.

5.1.3.5 Mixture component splitting

At this stage the state output pdfs consist of single full-covariance Gaussian pdfs. It has been shown that GMMs result in better recognition results than single Gaussians. To reach the full potential of context-dependent models, it is therefore necessary to replace the single Gaussian state pdfs to GMM state pdfs. The ideal scenario is to use GMM state pdfs instead of single Gaussian state pdfs when growing the decision trees. In order to grow GMM-based decision trees it is necesary to derive similar expressions to Eqs. 4.4 and 4.15, for respectively the log-likehood of a GMM and the total change in log-likelihood $\Delta\mathcal{L}(s,q)$. Furthermore, there will be no guarantee that the underlying mixtures of the GMMs will be assigned sufficient data in order to estimate reliable parameters. As a result of the increased complexity of GMM-based decision trees, the decision trees are grown using single Gaussian state pdfs. The single Gaussian state pdfs are subsequently transformed into GMM state pdfs. There are two methods to do this: a heuristic method and a data-driven method. It was decided that the data-driven method will be used. This requires the computation of the state-to-speech alignments using the Viterbi algorithm and the single Gaussian, parameter sharing, context-dependent models. Once the state-to-speech alignments have been determined, all the data corresponding to a shared state pdf is collected and divided into $\mathcal{M}$ clusters using vector quantisation. The means of each of the $\mathcal{M}$ clusters are used as the means of the new $\mathcal{M}$-mixture GMM that replaces the shared state Gaussian pdf. The covariance of each mixture component is set to a predefined covariance matrix. Once the parameter-shared context-dependent GMM models have

been constructed, it is once again necessary to re-estimate the model parameters using the normal re-estimation algorithms followed by embedded re-estimation.

5.2 Summary

This chapter outlined the implementation of the theory of the previous chapters. At first the necessity of labelled speech utterances (referred to as transcriptions) was explained. This was followed by a discussion on the different types of transcriptions, and the implications it has on the training method employed to construct acoustic-phonetic models. In order to initialise the acoustic phonetic models, it was necessary to have complete phonemic transcriptions. If these transcriptions were not available for a speech corpus, as was the case for the AST speech corpora, it was explained how the complete transcriptions can be created using *forced alignment* and models cloned from a similar speech corpus, such as the NTIMIT speech corpus.

The procedure to construct context-independent phoneme models was explained next. Once context-independent phoneme models have been constructed, they are used to construct the context-dependent phoneme models.

The procedure to construct context-independent phoneme models involves the gathering of sufficient statistics to build decision trees, which determine how to share the parameters of state pdfs of context-dependent phonemes derived from the same context-independent phonemes. In order to make the decision tree-based state clustering technique computationally feasible, it is limited to single, full-covariance Gaussian state output pdfs. However, it has been shown that mixture Gaussian distributions outperform single Gaussians and therefore it is necessary to replace the single Gaussian state pdfs with GMM state pdfs. This is accomplished by using a data-driven mixture component splitting technique, which requires the state-to-speech alignments and utilises vector quantistation.

Chapter 6

Experimental investigation

One purpose of this thesis is to create South African English phonemes models from the newly created South African English speech corpus. The speech corpus is the first professionally gathered database of South African English and this thesis represents the first phoneme recognition research performed on this database. Since no prior research on this database exists, it is necessary to establish baseline recognition results on a different speech corpus to which this research can be compared. It was decided to establish the baseline results on the NTIMIT speech corpus, because it is a mature database that has weathered much research. Furthermore, a large number of mistakes in the transcriptions of the NTIMIT speech corpus have been corrected over the years, which makes it very suitable for establishing baseline phoneme recognition results.

In order to obtain acceptable results on South African English it is necessary to evaluate the different phoneme modelling techniques discussed in the previous chapters. The purpose of this chapter is to report on the experiments performed to establish the baseline results and the results obtained on the South African speech corpora. Three different aspects of phoneme modelling are examined here. The first aspect that is examined is the different signal processing techniques, i.e. the acoustic processor or front-end of the speech recogniser. The second aspect examined is the acoustic models, specifically the influence of the statistical model complexity on the phoneme recognition. Thirdly, decision tree-based state clustering is used to construct context-dependent acoustic models. The context-dependent acoustic models that are investigated are both left- and right-context biphones and triphones.

6.1 Signal processing

Motivation

The motivation of this set of experiments is to evaluate the current popular and available signal processing techniques in order to select the set of techniques that are best suited

for phoneme recognition. The signal processing techniques were divided into the following three groups: signal pre-processing, feature extraction, and feature post-processing. The use of the pre-processing techniques, such as preemphasis and power normalisation (to reduce effects caused by speaker and channel identity) has been well established. Because the interest is in speaker-independent phoneme recognition, it is assumed that no new insights will be gathered by performing experiments on the group of signal pre-processing techniques. Both preemphasis and power normalisation will be performed on the signal prior to the other signal processing techniques.

NTIMIT Database

All the signal processing experiments were performed on the DARPA NTIMIT acoustic-phonetic continuous speech corpus [22, 21] (which is described more fully in Appendix C.1). The NTIMIT speech corpus contains 6 300 sentences, 10 sentences spoken by each of 630 speakers from eight major dialect regions of the United States. There are three primary reasons for performing the experiments on NTIMIT speech corpus. The first is that the NTIMIT speech corpus is a well-documented database that has been in widespread use for the last 10 years, which means that most of the problems associated with the database have been resolved. Furthermore, the NTIMIT database has been used in a number of studies and is ideally suited to form baseline results to which the experiments based on the South African language can be compared. The second is that the NTIMIT speech corpus is the TIMIT speech corpus, which has been transmitted over a telephone network. Because the TIMIT speech was recorded under controlled conditions using a high-quality microphone, it stands to reason that the NTIMIT speech corpus will be near-ideal speech under telephone conditions. The last reason is that the TIMIT corpus has been fully transcribed on a phonetic level.

Format of speech data

The NTIMIT speech corpus is the TIMIT speech corpus transmitted over a telephone network. The TIMIT speech corpus is recorded under good, controlled conditions using a high-quality microphone. The speech is sampled at 16kHz and stored in the standard NIST Sphere format, using 16 bits/sample.

Data sets

The experiments are performed by using the reduced set of 39 English phonemes proposed by K.F. Lee [27] (shown in Table C.1). The NTIMIT corpus consists of three types of sentences, i.e. **sa**, **sx** and **si**. Each of the 630 speakers speak two **sa**, three **si** and five **sx** sentences. The **sa** sentences are the same for all speakers, the **sx** sentences are from a

list of 450 phonetically balanced sentences, and the **si** sentences are from a list of 1 890 phonetically diverse sentences. In order to eliminate unfair bias toward certain phonemes only the **sx** and **si** sentences are used. The NTIMIT speech data amounts to a total of 4.2 hours of data.

Training set

The training set consists of 3 696 sentences of the NTIMIT proposed training set. This amounts to a total of 3.14 hours of data which is consists of 2.62 hours of speech and 0.52 hours of noise.

Testing set

The testing set consists of the 1 344 **sx** and **si** sentences of the of NTIMIT proposed test set. This amounts to a total of 1.06 hours of data which consists of 0.88 hours of speech and 0.18 hours of noise.

Execution

The techniques are evaluated using isolated, context-independent phoneme recognition experiments. Because complete phonetic transcriptions are available for the NTIMIT speech corpus, it is not necessary to create them by using forced alignment. The 39 context-independent phonemes, including the silence model, are modelled as three-state, left-to-right, one-skip HMMs. Each HMM is initialised as explained in Section 5.1.2.1, using the *Viterbi* algorithm. No embedded re-estimation is performed as the capability did not exist in the software at the time of the experiments. When performing the phoneme recognition experiments no language grammar is assumed. It is assumed that each phoneme has equal probability of occurrence. The results are evaluated based on each technique's phoneme recognition accuracy.

6.1.1 Feature extraction

Motivation

There are a few commonly used feature extraction techniques such as LPCC, MFCC and PLP in automatic speech recognition. The purpose of the feature extraction experiments is to compare these three feature extraction techniques to find the technique best suited to phoneme recognition. The feature vectors extracted with these techniques, without any added feature post-processing, will be referred to as the *base* feature vectors.

Experimental setup

The speech is preemphasised using the preemphasis filter discussed in Section 2.2.1. Each utterance of speech is blocked into $20ms$ frames. Subsequent frames overlap by $10ms$. Each frame is windowed with an equal length Hamming function to reduce spectral leakage.

Both 12^{th}-order and 18^{th}-order LPCCs are extracted by using the algorithm explained in Appendix A.1. A 22-order LP filter is used to obtain the LP spectrum of each frame, which is converted to 12^{th}- and 18^{th}-order LPCCs respectively. The order of the LP filter might seem to be quite high but, according to the results of Psutka *et al.* the high order results in higher recognition accuracy.

MFCCs are extracted by using a bank of 22 triangular filters, equally spaced according to the *mel*-frequency scale. For each frame the energy of each filter is converted to respectively 12^{th}- and 18^{th}-order MFCCs by using the discrete cosine transform (as discussed in Appendix A.31).

When PLPs are extracted, the magnitude spectrum of each frame is modelled with 22 LP parameters, but the LP parameters are derived from a perceptually motivated power spectrum (as discussed in Appendix A.37). The 22 LP parameters are then converted to 12^{th}- and 18^{th}-order PLPs, using the same conversion as LPCCs.

For each of the three different feature extraction techniques, a set of phoneme HMMs is trained. The phoneme HMMs respectively use 12^{th}- and 18^{th}-dimensional pdfs on each state for the 12^{th}- and 18^{th}-order extracted base feature vectors. For the purpose of comparing the base feature vectors it was decided to keep all variables of the experiment as simple as possible. Therefore, it was decided to use diagonal covariance Gaussian pdfs as the HMM state output pdfs. Three iterations of the *Viterbi* algorithm are used to re-estimate the HMM parameters.

The phoneme HMMs of each feature extraction technique are used to create parallel-HMM, isolated phoneme recognisers, which are used to evaluate the performance of each feature extraction technique. When performing isolated phoneme recognition, the *a priori* probability of each phoneme is assumed to be equal.

Results

The results of isolated phoneme recognition for the three base feature extraction techniqaues are tabulated as follows:

Table 6.1: *Isolated phoneme recognition results of base feature extraction techniques on the NTIMIT speech corpus.*

Features	Acc.
12^{th} order LPCC	30.92%
18^{th} order LPCC	33.88%
12^{th} order MFCC	31.50%
18^{th} order MFCC	29.81%
12^{th} order PLP	34.34%
18^{th} order PLP	34.00%

Interpretation

The results indicate that the PLP base feature extraction technique, without any feature post-processing, outperforms the other two techniques. It is interesting to note that 12^{th}-order features result in better performance than the 18^{th}-order features. It is important to realise that the 18^{th}-order features requires 18^{th}-dimensional pdfs. Therefore, there are more parameters that need to be estimated. To properly compare the difference in feature order one would somehow need to equalise the number of parameters.

6.1.2 Feature post-processing

Motivation

There might be information useful for discriminating phonemes embedded in the base feature vectors that can be extracted by post-processing the feature vectors. The three types of post-processing techniques examined are cepstral mean subtraction, velocity and acceleration, and linear discriminant analysis.

Experimental setup

The base feature vectors, which were extracted in the previous experiment, are augmented with the post-processing techniques. The influence of the post-processing on the recognition accuracy, robustness, data scarcity and computational requirement are measured subsequently. The computational requirement is measured by measuring the total number of parameters[1] of a given parallel HMM or spotter HMM.

[1]The total number of parameters is calculated as the sum of the total number of transition probabilities and the total number of free parameters of the state distributions.

6.1.2.1 Velocity and acceleration

Motivation

In statistical speech recognition it is usually assumed that the feature vectors are conditionally independent. This is done in an attempt to ensure that the mathematics is manageable. This assumption generally does not hold in practice, because the vocal apparatus forces continuity between successive spectral estimates of the speech signal [9]. One method of mitigating this strong assumption is the use of velocity and acceleration vectors. The velocity and acceleration vectors are respectively the first- and second-order temporal derivatives of the base feature vector sets and hence are commonly referred to as (Δ)s and ($\Delta\Delta$)s.

Experimental setup

The experimental setup used to examine the base feature vectors are used again. For this experiment, however, only the results on 12^{th}-order MFCC features are reported. The MFCC base feature vectors are augmented first with only velocity (Δs) and then with both velocity (Δs) and acceleration ($\Delta\Delta$s). The Δs are computed by using Eq. 2.17 on the base feature vectors and the $\Delta\Delta$s are computed by using Eq. 2.17 on the computed Δs.

For each of the augmented features, a set of phoneme HMMs is trained with three iterations of the *Viterbi* algorithm. In order to determine the influence of velocity and acceleration features on more complex state output pdfs, each set of 39 phonemes are modelled with different state output pdfs.

Hereafter, these phoneme HMMs are used to create parallel-HMM, isolated phoneme recognisers, which are used to evaluate the use of velocity and acceleration features. When performing isolated phoneme recognition, the *a priori* probability of each phoneme is assumed to be equal.

Results

The results of isolated phoneme recognition, when augmenting the 12^{th}-order MFCC base features with velocity and acceleration, are tabulated in Table D.3 and shown graphically in Fig. 6.1.

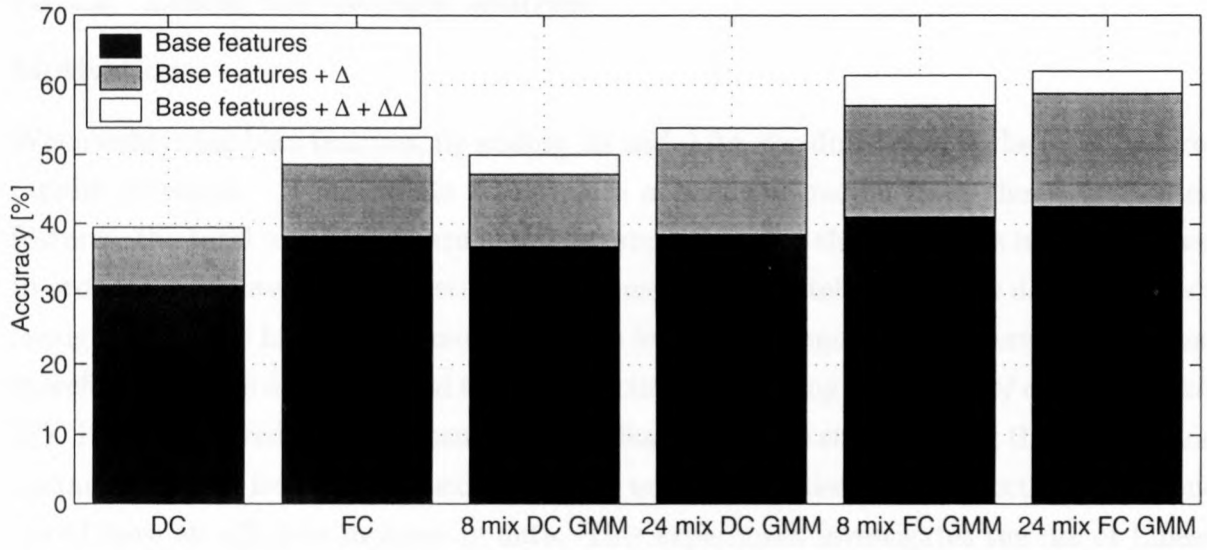

Figure 6.1: *Isolated phoneme recognition results of* 12^{th} *order MFCCs, augmented velocity and acceleration features, on the NTIMIT speech corpus.*

Interpretation

The augmentation of the base feature vectors with velocity features significantly improves the isolated recognition accuracy. Augmenting the base features with velocity and acceleration features improves the isolated recognition accuracy even further. The improvement due to the addition of acceleration features is not as much as the improvement due to the addition of velocity features. The conclusion can be made that the law of diminishing returns is applicable to the addition of higher-order time derivatives of base feature vectors (i.e. less information is extracted from the set of base feature vectors as the order of the time derivative(s) increases). The disadvantage of augmenting the base features is the increase in the dimension of the augmented feature vectors. This causes an increase in the computational requirement needed to estimate and recognise the phoneme models based on the higher-dimensional feature vectors, as well as an increase in the amount of data needed to obtain reliable estimates of the phoneme model parameters.

It is interesting to note that the more complex distributions are able to better utilise the extra information extracted with velocity and acceleration features. This is supported by the fact that the distributions with the least number of free parameters (i.e. DC Gaussians) have the lowest improvement in accuracy. However, when the number of parameters become very large, as in the case of 24 mixture FC GMMs, the improvements start to decline, because the number of parameters is more than the amount that can be reliably estimated from the data.

6.1.2.2 Linear discriminant analysis

Motivation

When enhancing base features, by adding Δs and $\Delta\Delta$s, the dimension of the total features rapidly increases. If one wants to estimate a phoneme model from these augmented features, the total number of parameters is proportional to the dimension of the features. In order to guarantee that the estimated parameters accurately model the data, one needs enough data. The higher the dimension of the features, the more parameters there are and therefore the more data is needed to obtain accurate modelling (*the curse of dimensionality* [6]). If one could reduce the dimension of the features, while still retaining the information contained in the features, the existing data would be utilised more effectively and one would have an *effective* increase in data. This experiment investigates the use of Linear Discriminant Analysis (LDA)[10, 25], which is similar to Fischer discriminants, to reduce the dimension of feature vectors.

Experimental setup

The base features, extracted in the feature extraction experiment (6.1.1), are used to perform the experiment. These base features are then augmented with velocity and acceleration features. Only the results on 12^{th}-order MFCC features are reported – as in the case with the velocity and acceleration features. The amount of dimension reduction is determined by a dimension retention factor, whereby the resultant dimension depends on the sum of the retained eigenvalues relative to the sum of all the eigenvalues of the eigenvalue matrix $\Omega_{\text{interclass}}$ (refer to Section 2.4.4). If the eigenvalue of dimension can be used as an indication of the amount of information in that dimension, the dimension retention factor corresponds to the amount of information that is retained. The dimension retention factor does not indicate the percentage of total dimensions that is retained.

For each set of features, a set of phoneme HMMs is trained with three iterations of the *Viterbi* algorithm. As the primary interest is to determine whether LDA can successfully retain information while reducing the dimension of the features, each set of 39 phonemes are modelled using a single diagonal-covariance Gaussian as HMM state output pdf. These phoneme HMMs are then used to create parallel-HMM, isolated phoneme recognisers, which are used to evaluate the performance of LDA. When performing isolated phoneme recognition, the *a priori* probability of each phoneme is assumed to be equal. The use of LDA as a post-processing technique is evaluated on the amount of dimension reduction and the retention of recognition accuracy.

Results

Table 6.2: *Influence of linear discriminative analysis on isolated phoneme recognition performed on NTIMIT, using* 12^{th}*-order MFCCs and DC-Gaussian pdfs.*

Δ	ΔΔ	LDA	Pdf.	Dim.	Acc.	Rel. Improv.	Dim. Reduc.
			DC-Gauss	12	31.14%	-	-
		√	DC-Gauss	6	31.22%	+0.26%	50.00%
√			DC-Gauss	24	37.76%	-	-
√		√	DC-Gauss	13	39.96%	+5.82%	45.83%
√	√		DC-Gauss	36	39.44%	-	-
√	√	√	DC-Gauss	17	42.47%	+7.75%	52.78%

Table 6.3: *Influence of linear discriminative analysis on isolated phoneme recognition performed on NTIMIT, using* 12^{th}*-order MFCCs and FC-Gaussian pdfs.*

Δ	ΔΔ	LDA	Pdf.	Dim.	Acc.	Rel. Improv.	Dim. Reduc.
			FC-Gauss	12	38.17%	-	-
		√	FC-Gauss	6	34.36%	-9.98%	50.00%
		√	FC-Gauss	7	35.47%	-7.07%	41.67%
		√	FC-Gauss	9	36.49%	-4.40%	25.00%
		√	FC-Gauss	10	37.29%	-2.31%	16.67%
		√	FC-Gauss	12	38.17%	0.00%	0.00%
√			FC-Gauss	24	48.58%	-	-
√		√	FC-Gauss	13	45.64%	-6.05%	45.83%
√		√	FC-Gauss	15	46.52%	-4.24%	37.50%
√		√	FC-Gauss	17	47.34%	-2.55%	29.17%
√		√	FC-Gauss	20	48.35%	-0.47%	20.00%
√		√	FC-Gauss	24	48.58%	0.00%	0.00%
√	√		FC-Gauss	36	52.48%	-	-
√	√	√	FC-Gauss	17	49.37%	-5.93%	52.78%
√	√	√	FC-Gauss	20	50.23%	-4.29%	44.44%
√	√	√	FC-Gauss	24	51.03%	-2.76%	33.33%
√	√	√	FC-Gauss	27	51.66%	-1.56%	25.00%
√	√	√	FC-Gauss	36	52.48%	0.00%	0.00%

Interpretation

The results for DC-Gaussian pdfs are shown in Table 6.2 and indicate that the use of LDA reduces the dimension of the features by up to 52.78%, while managing to improve the isolated phoneme recognition accuracy. When augmenting the base features with velocity and accelelation features, the recognition accuracy is actually improved with the use of LDA. The recognition improvements associated with the addition of velocity and acceleration features can now be utilised without the disadvantage of the increase in the dimension of the feature vectors. The dimension reduction application of LDA ensures this.

When full-covariance Gaussian pdfs are used, the results (shown in Table 6.3) indicate that the dimension reduction aspect of LDA reduces the isolated phoneme recognition accuracy. It is interesting to note that the influence of the dimension reduction is reduced when the base feature vectors are augmented by velocity and acceleration features. For the worst result of fully-augmented feature vectors, a 52.78% reduction in dimension leads to a relative decrease of only 5.93%. When LDA is applied, without reducing the dimension of the resultant feature vectors, the isolated phoneme recognition accuracy is exactly the same as when LDA is not applied at all.

The increase and decrease in recognition accuracy for respectively DC-Gaussian and FC-Gaussian pdfs can be readily understood if one examines the histogram of correlation coefficients before and after LDA (shown in Figs. 2.8 and 2.9). It is clear that after LDA, the correlation coefficients are much lower than before LDA. Therefore, the dimensions of the feature vectors are less correlated and DC-Gaussian pdfs are better able to model the feature vectors. The loss in accuracy due to the reduction in information is consequently less than the increase in accuracy due to the dimensions being less correlated. However, the correlation coefficients are not zero. Seeing as FC-Gaussian pdfs gain nothing by decorrelating the dimensions of the feature vectors, the reduction in dimension degrades the recognition accuracy of FC-Gaussian pdfs.

Based on the FC-Gaussian and DC-Gaussian results, a dimension retention factor of 98.2% will be used for the subsequent experiments. For MFCCs that have been augmented with velocity and acceleration features, LDA reduces the dimension for 36 to 17 dimensional features.

6.1.2.3 Cepstral mean subtraction

Motivation

Varying channel characteristics result in convolutional noise in the speech signal. This is transformed into additive noise in the cepstral domain causing unwanted bias in the extracted feature vectors. By subtracting the mean of the cepstral coefficients, the bias is

removed and one can compensate partially for the effects of the room and the microphone. This experiments measure the influence of CMS on telephone speech.

Experimental setup

The previous experiments demonstrated the advantages of using velocity and acceleration features, combined with LDA. The purpose of CMS is to compensate for noise in order to improve recognition accuracy. As complex statistical pdfs generally have high recognition accuracies, one wishes to determine whether CMS still improves recognition accuracy when very complex statistical pdfs are used as HMM state output pdfs.

The base features, augmented with velocity and acceleration features and dimensionally reduced with LDA, are used to conduct these experiments. An experiment is performed for each of the base feature extraction techniques, with and without CMS post-processing.

For each experiment, a set of 39 phonemes are modelled with 24 mixture Gaussian pdfs as HMM state output pdfs. The parameters of the HMMs are estimated using three iterations of the *Viterbi* algorithm. These sets of phoneme HMMs are then used to create parallel-HMM, isolated phoneme recognisers, which are used to evaluate the use of CMS. When performing isolated phoneme recognition, the *a priori* probability of each phoneme is assumed to be equal.

Results

The results of isolated recognition, with and without CMS, are tabulated in Table 6.4.

Interpretation

For all the different feature extraction techniques examined in this thesis, the use of cepstral mean subtraction marginally increases the isolated phoneme recognition accuracy. It is therefore used as a post-processing step in the rest of the phoneme recognition experiments.

6.1.3 Summary

The results of the signal-processing experiments are summarised as follows:

- For isolated phoneme recognition the PLP feature extraction technique results in the highest recognition accuracy, closely followed by MFCCs. As most modern ASR systems utilise MFCCs as the feature extraction technique and most research done in LVCSR utilises MFCCs, MFCCs will be used in all subsequent experiments to make the results of this thesis easily comparable to other research.

Table 6.4: *Recognition accuracies on NTIMIT using CMS.*

Features	CMS	Pdf.	Dim.	Acc.	Rel. Improv.
12^{th}-order LPCC		24 mixture FC GMM	13	55.12%	-
12^{th}-order LPCC	√	24 mixture FC GMM	13	55.79%	+1.22%
12^{th}-order MFCC		24 mixture FC GMM	17	59.31%	-
12^{th}-order MFCC	√	24 mixture FC GMM	17	60.35%	+1.75%
12^{th}-order PLP		24 mixture FC GMM	18	59.65%	-
12^{th}-order PLP	√	24 mixture FC GMM	18	61.11%	+2.45%
18^{th}-order LPCC		24 mixture DC GMM	17	54.70%	-
18^{th}-order LPCC	√	24 mixture DC GMM	17	54.83%	+0.24%
18^{th}-order LPCC		24 mixture FC GMM	17	59.92%	-
18^{th}-order LPCC	√	24 mixture FC GMM	17	60.24%	+0.53%
18^{th}-order MFCC		24 mixture DC GMM	24	56.32%	-
18^{th}-order MFCC	√	24 mixture DC GMM	18	58.76%	+4.33%
18^{th}-order MFCC		24 mixture FC GMM	24	59.82%	-
18^{th}-order MFCC	√	24 mixture FC GMM	18	60.64%	+1.37%
18^{th}-order PLP		24 mixture DC GMM	22	55.51%	-
18^{th}-order PLP	√	24 mixture DC GMM	22	56.52%	+1.82%
18^{th}-order PLP		24 mixture FC GMM	22	60.77%	-
18^{th}-order PLP	√	24 mixture FC GMM	22	61.83%	+1.74%

- The augmentation of base features with the velocity and acceleration features mitigates the conditionally independent assumption of HMMs and drastically increases the isolated phoneme recognition accuracy.

- For DC-Gaussian pdfs, the use of LDA as a dimension-reduction technique reduces the dimension of the feature vectors considerably, while having an insignificant influence on or improving the isolated phoneme recognition accuracy. When FC-Gaussian pdfs are used, LDA decreases the isolated phoneme recognition accuracy. However, the relative reduction in dimension is more than the relative decrease in accuracy. In order to reduce the computational requirement, LDA will be used in all subsequent experiments in order to reduce the dimension of base feature vectors, augmented with velocity and acceleration features. This will improve the recognition accuracy of experiments using DC-Gaussian pdfs, while decreasing the recognition accuracy of experiments using FC-Gaussian pdfs.

- Cepstral mean subtraction compensates for the noise and environmental variability associated with telephone speech. It will be used in all subsequent experiments.

6.2 Statistical modelling of HMM state distributions

Motivation

Typically increasing the statistical model complexity increases the recognition accuracy of a recogniser. However, the statistical complexity is constrained by the amount of available speech data and the computational requirement. The reference to statistical modelling complexity is made in the context of HMM-based speech recogniser use, and is specifically made in reference to the complexity of the HMM state output pdfs. This section is concerned with finding a balance between recognition accuracy and the computational efficiency.

Setup

The statistical modelling experiments are all performed on the DARPA NTIMIT speech corpus [22, 21]. As in the case of the signal processing experiments, the reason for using NTIMIT is to establish comparable baseline recognition results. The experiments are performed with the previously mentioned English phoneme set proposed by K.F. Lee (see Table C.1).

Execution

Different HMM state output probability distributions are examined, using isolated, context-independent phoneme recognition experiments. The pdfs examined are single and mixture Gaussian distributions. They are examined by using both diagonal and full covariance matrices. Because the NTIMIT speech corpus has complete transcriptions it is not necessary to create them using forced alignment. The 39 context-independent phonemes, including the silence model, are modelled as three-state, left-to-right, one-skip HMMs. Each HMM is initialised by using the *Viterbi* algorithm (see Section 5.1.2.1). No embedded re-estimation is performed because the capability did not exist at the time of the experiments. When performing the phoneme recognition experiments, no language grammar is assumed. It is assumed that each phoneme has equal probability of occurrence. The results are evaluated based on phoneme recognition accuracy.

Number of parameters vs. computational efficiency

In subsequent experiments, one basis on which the different phoneme modelling techniques are compared is computational efficiency. Computational efficiency could be measured by just measuring the recognition speed of each phoneme modelling technique. The problem with measuring recognition speed is that the assumption is made that the modelling

techniques are implemented in a computationally optimal fashion. In order to separate the modelling techniques from their implementation it was decided to use the total number of model parameters as an indication of computational efficiency. In this section it is motivated that the total number of model parameters can be used as an indication of computational efficiency. The influence of an increase in the parameters of the HMM state transitions and the HMM state distributions will be considered separately.

HMM state transitions

The *Viterbi* algorithm is used for decoding. The order of the Viterbi algorithm is

$$O(NLT), \quad \text{where} \quad \begin{array}{l} \text{N is the number of HMM states,} \\ \text{L is the average number of transition parameters per state} \\ \text{T is the length of the observation sequence.} \end{array} \tag{6.1}$$

The parameter T will be constant if the same testing set is used for all experiments. The product NL is the total number of transition parameters of the HMM. Therefore the increase in the computional requirements of a larger HMM topology is proportional to the increase in the total number of transition parameters.

HMM state distributions

The computational requirements of computing the log-likelihood of a D-dimensional pdf is investigate for single Gaussian pdfs and GMMs. To compute the log-likelihood of a diagonal-covariance Gaussian pdf requires $2(D-1)$ summations and $2D$ multiplications. The log-likelihood computation therefore requires $4D-2$ operations. A diagonal-covariance Gaussian pdf has $2D$ parameters and therefore the log-likelihood computation is proportional to the number of model parameters.

To compute the log-likelihood of a full-covariance Gaussian pdf requires $\frac{(D+4)(D-1)}{2}$ summations and $\frac{D(D+3)}{2}$ multiplications. The log-likelihood computation therefore requires $D(D+3)-2$ operations. A full-covariance Gaussian pdf has $\frac{D(D+3)}{2}$ parameters. If D becomes large then the -2 term becomes negligible and therefore the log-likelihood computation is proportional to the number of model parameters.

To compute the log-likelihood of a M mixture GMM requires the computation of the log-likelihoods of M pdfs, M exponentials and $M-1$ summations. The log-likelihood computation is therefore proportional to $M \times$ (Log-likelihood computation of pdf).

A M_{DC}-mixture DC GMM has $M_{DC}\,[2D+1]$ model parameters and the log-likelihood computation is proportional to $M_{DC}\,[2D-2]$. If the influence of the constants are assumed to be negligible, then the log-likelihood computation is proportional to the number of model parameters.

A M_{FC}-mixture FC GMM has $M_{FC}\left[\frac{D(D+3)}{2}+1\right]$ model parameters and the log-likelihood computation is proportional to $M_{FC}\left[D(D+3)-2\right]$. If the influence of the constants are assumed to be negligible, then the log-likelihood computation is proportional to the number of model parameters.

6.2.1 Gaussian mixture state probability distributions

Motivation

When using single Gaussian pdfs as HMM state output pdfs the inherent assumption is that the distribution of the speech in the feature space is Gaussian in nature. This assumption does not necessarily hold true for speech. Gaussian Mixture Models (GMMs), or mixture Gaussian distributions, might be able to better model the distribution of the speech in the feature space. As the number of mixtures increase, the data necessary to estimate reliable parameters, as well as the computational requirement, increases. The optimal number of mixtures to use when modelling speech is therefore not readily apparent, as it includes conflicting factors such as recognition accuracy, data scarcity, and computational requirement. The purpose of this experiment is to empirically determine the optimal number of mixtures needed to model the speech features.

Experimental setup

The speech is preemphasised using the preemphasis filter discussed in Section 2.2.1. Each utterance of speech is blocked into $20ms$ frames and subsequent frames overlap by $10ms$. Each frame is windowed with an equal length Hamming function to reduce spectral leakage at the ends of each frame.

It was decided to use 12^{th}-order MFCC features, as the results for PLP and MFCC feature extraction are very similar. Another reason is that the proposed speech recogniser, that will be built based on the South African English phonemes, will use MFCCs as features. The MFCCs are post-processed with CMS, augmented with velocity and acceleration features, and dimensionally reduced using LDA. For each of the experiments a set of phoneme HMMs is trained, using three iterations of the *Viterbi* algorithm. These sets of phoneme HMMs are then used to create parallel-HMM, isolated phoneme recognisers, which are used to determine the optimal number of mixtures.

For the HMM state output pdfs both diagonal and full-covariance GMMs are used. The only parameter which changes between successive experiments is the number of mixtures in the state output GMMs.

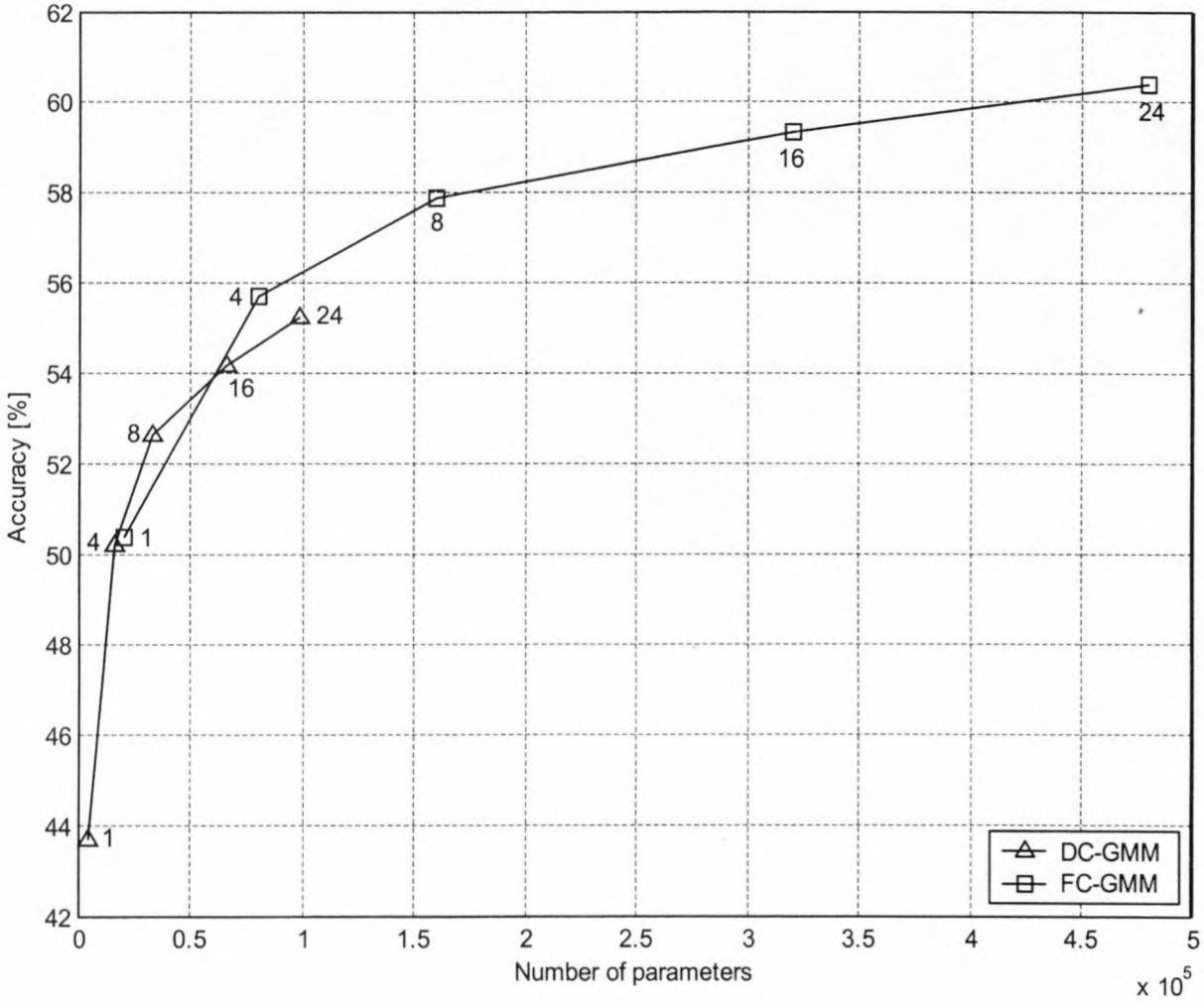

Figure 6.2: *Isolated phoneme recognition results on NTIMIT for different number of mixtures of GMM state output probability distributions.*

Results

The isolated phoneme recognition results on NTIMIT for different number of mixtures are tabulated in Table D.4. In Fig. 6.2 the accuracies are plotted with respect to the computational requirement as measured by the total number of parameters.

Interpretation

The more complex the models become, the higher the isolated phoneme recognition accuracies tend to be. However, the increase in recognition accuracy is not in a linear relationship with the increase in model complexity. It can clearly be seen in Fig. 6.2 that the relative increase in accuracy diminishes as the model complexity increases. Furthermore, the more complex the phoneme model, the higher the recognition computational requirement. It is interesting to note that, for four to 24 mixture DC GMMs, the recognition accuracy is higher than that for the FC GMMs having the same computational

requirement.

6.2.2 Diagonal covariance vs. full covariance

Motivation

Gaussian pdfs using full-covariance matrices are much more powerful in modelling inter-dimensional dependencies, but take long to calculate and require more data to obtain reliable parameter estimates than Gaussian pdfs using other types of covariance matrices. A single full covariance Gaussian has $\frac{D(D+3)}{2}$ free parameters to estimate (where D is the dimension of the matrix) and requires an absolute minimum of D feature vectors to obtain a reliable estimate. Single diagonal-covariance Gaussians are much less flexible when modelling distributions, but require much less data to estimate. A single diagonal-covariance Gaussian has only $2D$ free parameters and requires an absolute minimum of two feature vectors to obtain a computable estimate. It has been shown before that full-covariance Gaussians can be approximated with diagonal covariance GMMs. The purpose of this experiment is to perform a comparison between diagonal- and full-covariance GMMs, based on an equal number of parameters and best recognition basis. A M_{FC} mixture full-covariance GMM has $M_{FC}\left[\frac{D(D+3)}{2}+1\right]$ free parameters, while a M_{DC} mixture diagonal-covariance has $M_{DC}\left[2D+1\right]$ free parameters.

Experimental setup

As with the previous statistical modelling experiment, only the results of 12^{th}-order MFCCs features are reported. The MFCC features are post-processed with CMS, augmented with velocity and acceleration features, and dimensionally reduced using LDA.

Two sets of phoneme HMMs are created for each experiment. The first set uses full-covariance GMMs as HMM state output pdfs and the second uses diagonal-covariance GMMs. The number of mixtures is chosen so that the FC GMM and the DC GMM has a nearly equal number of free parameters. Each set of phoneme HMMs is trained using three iterations of the *Viterbi* algorithm. These sets of phoneme HMMs are then used to create parallel-HMM, isolated phoneme recognisers, which are used to compare the different types of GMMs.

Results

The results of the experiment are tabulated in Table D.5. In Fig. 6.3 the accuracies are plotted with respect to the computational requirement as measured by the total number of parameters.

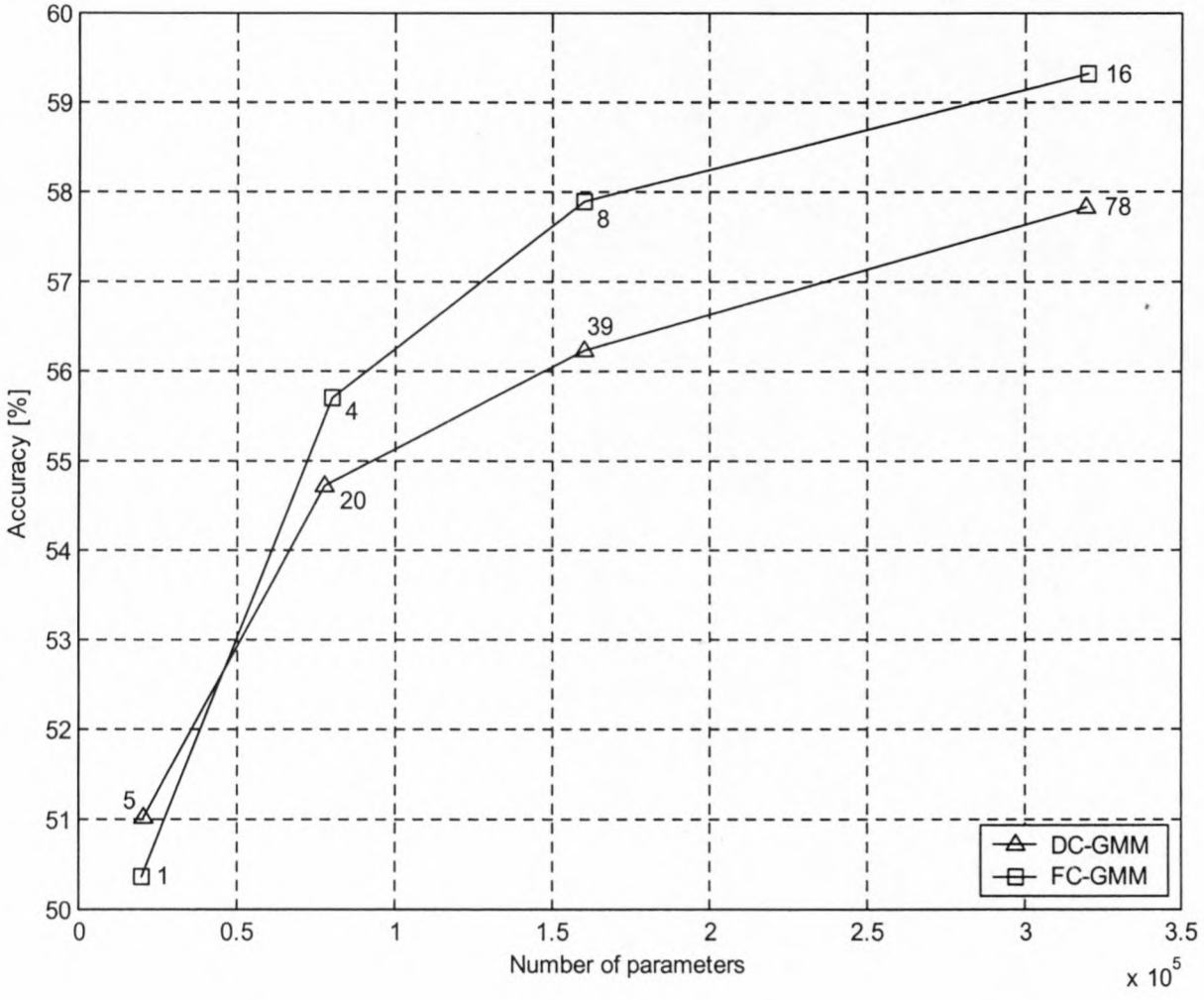

Figure 6.3: *Isolated phoneme recognition results on NTIMIT comparing diagonal and full covariance state pdfs with the nearly equal number of parameters.*

Interpretation

It is usually assumed that the correlation among feature coefficients is weak for MFCCs [24, p. 402]. Therefore, diagonal-covariance matrices are sufficient to model MFCC features. Most LVCSR systems tend to use MFCCs (Cambridge's HTK [31, p. 114], CMU's JANUS [45], Microsoft's Whisper [24, p. 463]) and diagonal-covariance GMMs for the state output pdfs of their phoneme models in order to smooth the covariance matrices and to reduce the computational requirement [26, p. 32]. The results of this experiment show that for isolated phoneme recognition, diagonal-covariance-based GMMs with five mixtures or less outperform full-covariance based GMMs. This is consistent with the assumption that DC-Gaussian pdfs adequately model MFCCS. However, somewhere between 5 and 20 mixture DC GMMs, full-covariance based GMMs start to perform better than DC GMMs. A possible explanation is that, when the number of DC GMM mixtures becomes very large, there might not be enough training data to reliable estimate

each individual DC-Gaussian mixture. Although DC GMM and FC GMM have an equal number of parameters, the FC GMM has a fewer number of mixtures and might therefore be able to more effectively utilise the training data. Fig. 6.3 indicates that the use of full-covariance based GMMs, to obtain acceptable recognition accuracies without sacrificing recognition speed, can be justified when using MFCCs.

6.2.3 Summary

The results for the statistical modelling experiments are summarised as follows:

- As expected, the isolated phoneme recognition accuracy improves as the complexity of the state distribution increases. The improvement in accuracy is not linearly related to the complexity, as the improvement saturates when the model becomes too complex to be reliably estimated from the limited amount of data.

- For an equal number of mixtures, phoneme models that use diagonal-covariance GMM state distributions generally has a lower computational requirement than phoneme models that use full-covariance GMM state distributions. The lower computational requirement of DC GMM state distributions comes at the cost of a lower recognition accuracy. This is to be expected, as the FC GMM state distributions have more parameters than the DC GMM state distributions, and are therefore better able to model the data.

- It is generally accepted that, in order to obtain faster recognition speeds, it is better to use DC GMM than FC GMM state distributions. The reasoning is that any FC GMM distribution can be modelled equally well using a DC GMM distribution with a sufficient number of mixture distributions. If the total number of parameters is used as an indication of the recognition speed then, by comparing DC GMM and FC GMM state distributions on an equal-parameter basis, it can be assumed that they will have similar recognition speeds. The equal-parameter comparison shows that there are areas where FC GMMs have a higher recognition accuracy than DC GMMs. Because DC GMM based recognition is improved by the application of LDA and FC GMM based recognition is degraded, there is even more grounds for the use of FC GMM based phoneme recognition.

6.3 Context-dependent modelling

Motivation

As mentioned previously in Chapter 4, arbitrary word models can be constructed when using context-independent phoneme models, but the recognition accuracy is not very high. The reason for this poor performance of context-independent phonemes is called *co-articulation*: the fact that phonemes are influenced by the context in which they appear. Research has shown that context-dependent phonemes (i.e. triphones and biphones) model the context dependency of phonemes more accurately. The experiments reported in this section evaluates some of the different types of context-dependent phonemes, i.e. left- and right context biphones and triphones.

Setup

The context-dependent modelling experiments are performed on the NTIMIT speech corpus. The primary purpose is to create baseline recognition results to which the context-dependent recognition results on the South African databases can be compared. A secondary reason is that numerous context-dependent modelling experiments have been performed on NTIMIT. This makes it useful to test the parameter sharing code which was written as part of this research.

All the experiments on NTIMIT are performed with the previously mentioned English phoneme set proposed by K.F. Lee (see Table C.1).

Execution

The different types of context-dependent phoneme models are examined by using isolated and continuous phoneme recognition experiments. Because the NTIMIT speech corpus has time-aligned transcriptions it is not necessary to create them using forced alignment.

The transcriptions of the NTIMIT corpus contain complete phonetic transcriptions that do not contain the word boundaries. The complete word transcriptions contain the word boundaries and it is possible to insert the word boundaries into the phonetic transcriptions. According to [17] a word recognition system using cross word context dependency can lead to increased system accuracy. Therefore, the decision was made to use cross-word context dependency to model the context-dependent phonemes in order to obtain increased system accuracy and to make the experiments more comparable to the results of [26, 27, 52]. For each of the different types of context-dependent phonemes, the cross-word, context-dependent phonemes that occur in the training set and in the full database is determined. Each of the phonemes, including the silence model, are modelled as three-state, left-to-right, one-skip HMMs. Each HMM is initialised as explained

in Section 5.1.3.1. Because of the limited amount of data it is necessary to share the parameters of context-dependent phonemes derived from the same context-independent phonemes. As it will be necessary to perform *embedded re-estimation* (Section 5.1.3.4) with the South African speech corpora, the context-dependent phonemes, based on the NTIMIT speech corpus, are also trained using embedded re-estimation after the parameter sharing has been performed. The primary reason is to make the context-dependent recognition results on the NTIMIT and South African databases comparable.

When performing the phoneme-recognition experiments no language grammar is assumed and it is assumed that each phoneme has equal probability of occurrence. The results are evaluated based on each context-dependent phoneme type's recognition accuracy and the speed of recognition.

6.3.1 Decision tree-based state clustering

Motivation

To obtain models with well-estimated parameters, one typically needs 100 or more examples of each triphone. To accurately model the distributions of real speech spectra, and thereby obtain acceptable recognition accuracies, it is necessary to have fairly complex output distributions for each HMM state output pdf. If one takes into account that the signal analysis and feature extraction techniques convert the one-dimensional speech signal into typically 10-20 dimensional feature vectors, the number of parameters associated with each model becomes quite large. For example, if a three-state, left-to-right HMM phoneme model, with eight-mixture, full-covariance Gaussian state distributions, are used to model 17-dimensional feature vectors, each triphone model has about 4 100 parameters. If one tries to model 17 000 triphones, 69.8 million parameters would have to be estimated. If 40 monophones are modelled, only 164 000 parameters need to be estimated. It can therefore be seen that monophones are usually well-estimated, but, due to a lack of context-modelling, they are too general to differentiate between different phonemes. Triphones, on the other hand, are more specific, but are usually poorly estimated due to a lack of sufficient data. Decision tree-based state clustering is a method used to alleviate the data scarcity problem. The HTK toolkit includes decision tree-based state clustering, but only for diagonal-covariance Gaussian distributions. Therefore, it was necessary to implement a decision tree-based state clustering algorithm that could also accommodate full-covariance Gaussian distributions. The purpose of this experiment is to show that real-world speech corpora have insufficient information to train context-dependent phoneme models without parameter sharing. Furthermore, the experiment shows that the implemented decision tree-based state clustering algorithm shares the data efficiently and results in acceptable recognition accuracies.

Experimental setup

The speech is preemphasised using the preemphasis filter discussed in Section 2.2.1. Each utterance of speech is blocked into $20ms$ frames and subsequent frames overlap by $10ms$. Each frame is windowed with an equal length Hamming function to reduce spectral leakage at the ends of each frame.

MFCCs are extracted by using a bank of 22 triangular filters equally spaced according to the *mel*-frequency scale. For each frame the energy of each filter is converted to respectively 12^{th}-order MFCCs by using the discrete cosine transform (as discussed in Appendix A.31). The features are post-processed with CMS, augmented with velocity and acceleration features, and dimensionally reduced by using LDA.

For each of the different types of context-dependent models, two sets of phoneme HMMs are created. On the one set no parameters are shared and on the other set decision tree-based state clustering has been performed to share the parameters of the HMM state output pdfs. The phoneme HMMs are modelled using full-covariance Gaussians as HMM state output pdfs.

Both isolated phoneme and continuous phoneme recognition experiments are performed to evaluate the performance of the context-dependent models with and without parameter sharing. No language model or grammar is assumed for the phoneme recognition experiments.

Results

The isolated phoneme recognition accuracies with and without parameter sharing is tabulated in Table 6.5.

Table 6.5: *Isolated phoneme recognition results on NTIMIT, comparing context-dependent phoneme models with and without parameter sharing (via decision tree-based state clustering).* 12^{th}*-order MFCCs are used as features.*

Modelling	Param. Sharing	Acc.	Params.	Rel. Improv.
Monophones		50.35%	20 202	-
LC Biphones		56.47%	656 240	-
LC Biphones	√	56.78%	223 760	+0.55%
RC Biphones		49.90%	656 240	-
RC Biphones	√	56.29%	224 100	+12.81%
Triphones		48.78%	7 960 160	-
Triphones	√	59.20%	494 440	+21.36%

Interpretation

It can clearly be seen from the results of Table 6.5 that the use of parameter sharing is highly recommended. For right-context biphones and triphones the recognition accuracy is improved, while the number of parameters is reduced considerably. The number of parameters for right-context biphones is reduced by a factor of 2.93, causing an equal reduction in the computational requirement and a relative improvement of 12.81% in recognition accuracy. Parameter sharing has the virtually no influence on the recognition accuracy of left-context biphones – this is unexpected and no suitable explanation could be found. However, the number of parameters was still reduced by a factor of 2.93. In the case of triphone models, the number of parameters was reduced by a factor of 16.10, causing a relative improvement of 21.36% in recognition accuracy.

6.3.2 Using biphones as context-dependent modelling unit

Motivation

Biphones are the set of context-dependent phonemes with the least context dependency. As mentioned previously, two types of biphones exist: left-context- (LC-biphone) and right-context dependent (RC-biphone).

The first question that this experiment seeks to answer is, "How does biphone phoneme recognisers compare to context-independent phoneme recognisers?" It is important to compare the different recognisers with two criteria. The first criterium is to compare the phoneme recognisers when they all have the same number of free parameters. This means that, although the biphones are more context-dependent, the HMM state output pdf will be less complex, because there are a large number of biphones. On the other hand, there are about 40 monophones in the English language, and therefore the HMM state output pdf of monophones can be much more complex.

The second criterium is the best accuracy irrespective of model complexity (within reason). The reason for this is the limited amount of available data. Although monophones are not context-dependent, monophones with more complex state pdfs can be reliably estimated, because there is more data available for a monophone, than there is for a biphone.

Experimental setup

As is the case with the previous experiment, the two different recognisers will be compared using 12^{th}-order MFCCs, which are post-processed by using CMS, augmented with velocity and acceleration features, and dimensionally reduced by using LDA.

Two sets of phonemes are trained on the NTIMIT speech corpus. The first is a set of left context-dependent biphones and the other is a set of right context-dependent

biphones. Decision tree-based state clustering is performed to share the state output pdf parameters across biphones derived from the same monophone. When constructing the decision trees a state occupancy of $\tau = 350$ is used as the minimum number of features vectors per state cluster (based on the context-dependent modelling research performed by Odell, Woodland and Young [31, 50, 51, 52]). The set of phonemes consists of the left- and right-context biphones found in the set of training data. It was decided to use the ideal case and to only construct the unseen biphones that occur in the full data set. In other words, the complete set of phonemes consists of all the biphones that occur in the full speech corpus. No biphones are modelled that does not occur in the speech corpus.

With each of these sets of phoneme HMMs, both parallel-HMMs and spotter-HMMs are constructed that are respectively used to perform isolated and continuous phoneme recognition. No language model or grammar is assumed for the phoneme recognition experiments, but paths in the spotter is constrained so that only phoneme sequences are allowed that coincide with the biphone modelling. The context-dependent phoneme

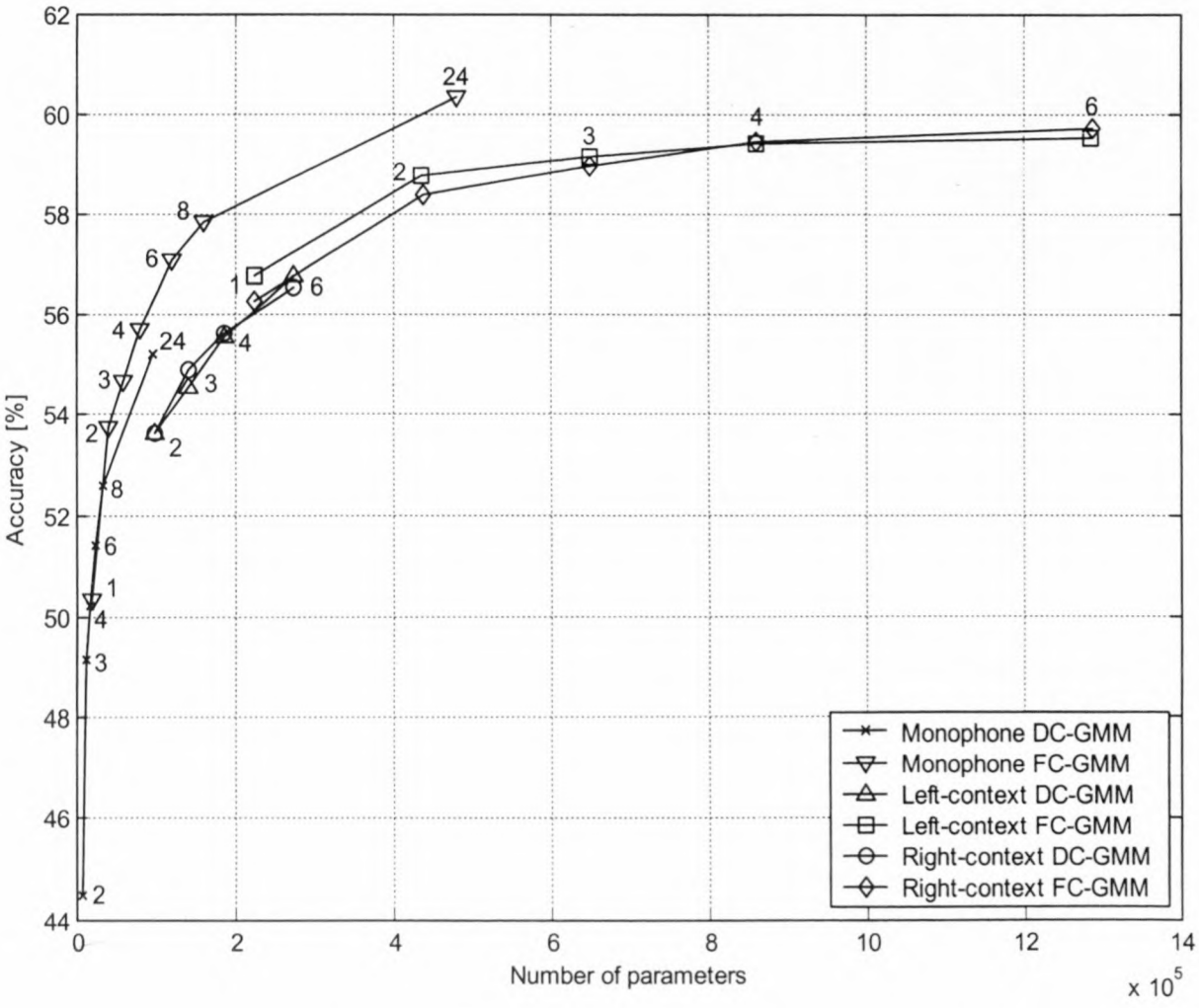

Figure 6.4: *Isolated phoneme recognition results on NTIMIT using monophones, left-context biphones, and right-context biphones to model context dependency.*

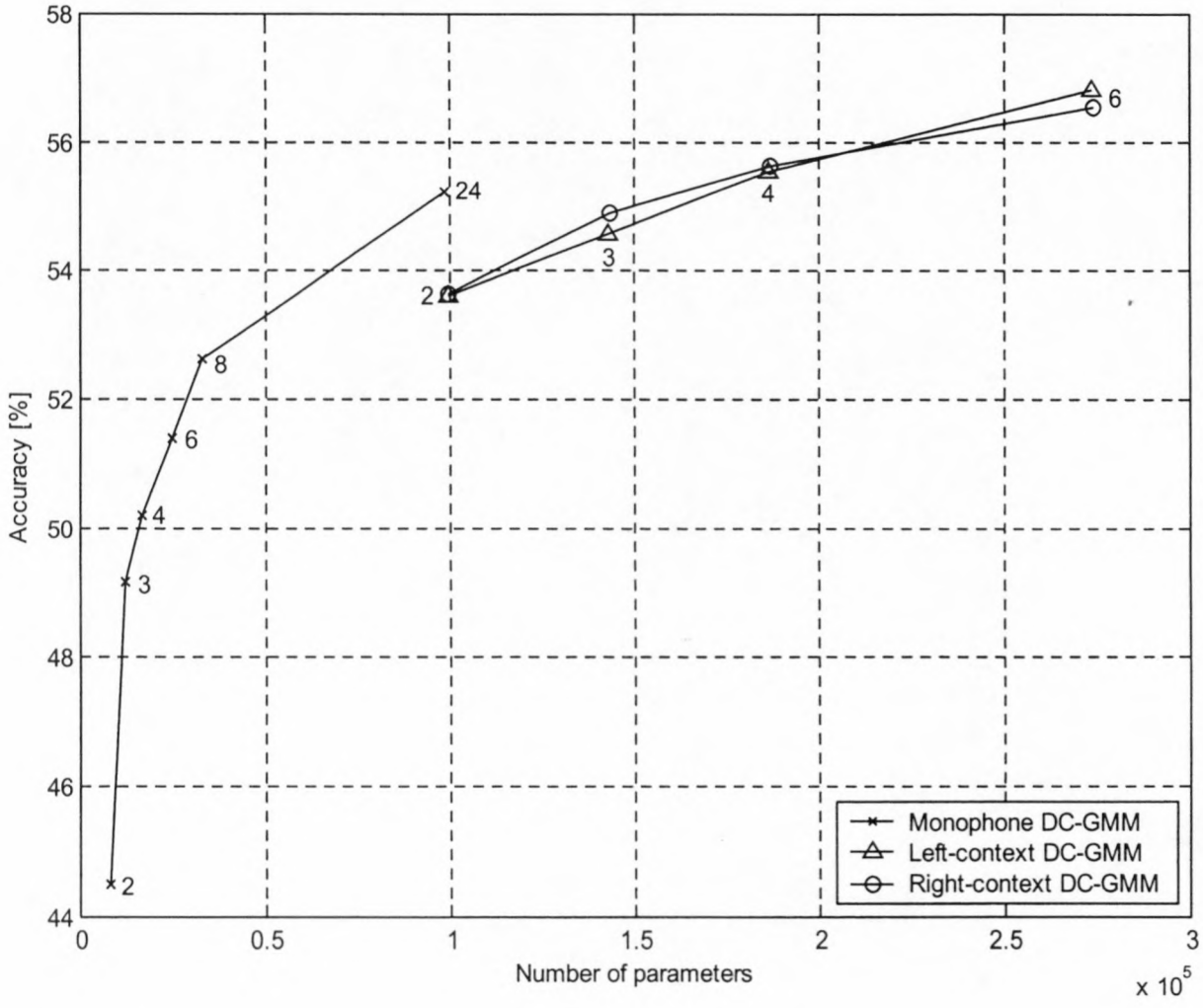

Figure 6.5: *Isolated phoneme recognition results on NTIMIT using diagonal covariance GMM monophones, left-context biphones, and right-context biphones to model context dependency.*

recognition results are then compared to the context-independent phoneme recognition results.

Results

The isolated phoneme recognition accuracies, using left- and right-context biphones, are tabulated in Table D.6. In Fig. 6.4 the recognition accuracies for monophones, and left- and right-context biphones are plotted with respect to the computational requirement as measured by the total number of parameters. Figs. 6.5 and 6.6 are subplots of Fig. 6.4 respectively, showing the DC GMM and FC GMM recognition results.

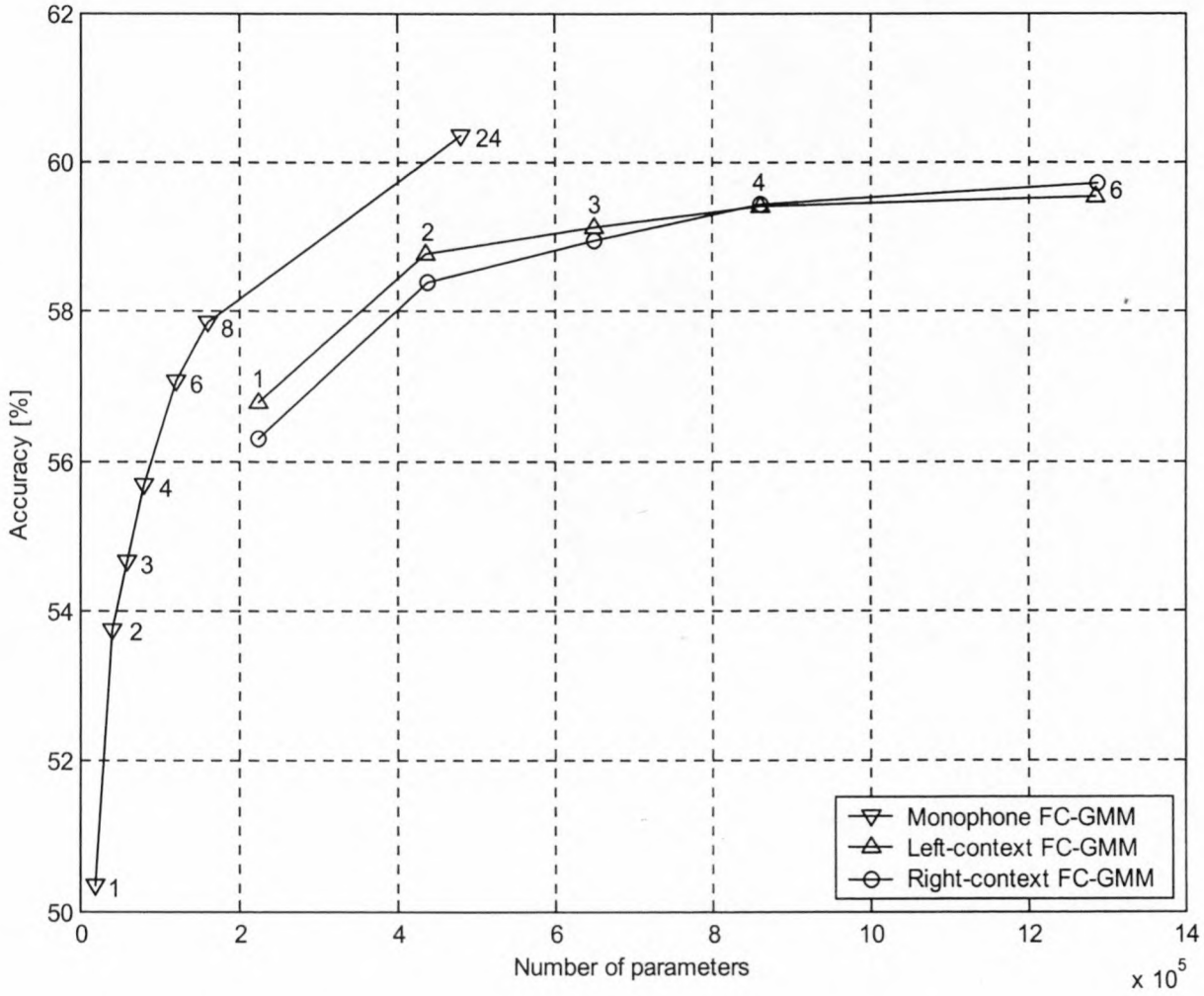

Figure 6.6: *Isolated phoneme recognition results on NTIMIT, using full covariance GMM monophones, left-context biphones, and right-context biphones to model context dependency.*

Interpretation

When performing isolated phoneme recognition, without any language model, the results seem to indicate that monophone models outperform left- and right-context biphone models. However, if the results for DC GMMs and FC GMMs (Figs. 6.5 and 6.6) are examined, it can be seen that an increase in the number of mixtures for DC GMMs causes an increase in recognition accuracy, while the recognition accuracy for FC GMMs were saturated after two mixtures. Therefore, an increase in the number of mixtures of the FC GMMs do not cause any significant increase in recognition accuracy. The most probable reason for this saturation is a lack of enough data to obtain proper estimates for the larger mixture FC-GMMs. A state occupancy of $\tau = 350$ was used as the minimum amount of data per shared distribution. While this may be enough data to estimate single mixture FC GMMs, it is probably not enough data when estimating larger mixture FC GMMs.

6.3.3 Using triphones as context-dependent modelling unit

Motivation

Triphones are even more context-dependent than biphones, separately modelling phonemes in context of their left and right neighbours. Unfortunately there are also considerably more triphones than biphones. The question that this experiment seeks to answer is, "What is gained in recognition accuracy but lost in computational efficiency by using triphones instead of biphones?"

Experimental setup

As is the case in the previous experiment, the triphone-based phoneme recogniser will be evaluated using 12^{th}-order MFCCs. The MFCCs are post-processed using CMS, augmented with velocity and acceleration features, and dimensionally reduced by using LDA.

A set of phonemes, consisting of all the triphones that occur in the training set, is constructed from the training data. Decision tree-based state clustering is performed to share the state output pdf parameters across triphones derived from the same monophone. When constructing the decision trees, state occupancies of respectively $\tau = 350$ and $\tau = 700$ are used as the minimum number of feature vectors of data per state cluster. It was decided to use the ideal case and only construct the models of unseen contexts that occur in the full data set (and not in the training data). In other words, the complete set of phonemes consist of all the triphones that occur in the full speech corpus and no triphones are modelled that does not occur in the speech corpus.

With the set of phoneme HMMs, both a parallel-HMM and a spotter-HMM are constructed, which are respectively used to perform isolated and continuous phoneme recognition. No language model or grammar is assumed for the phoneme recognition experiments, but paths in the spotter-HMM is constrained so that only phoneme sequences are allowed that coincide with the triphone modelling. The context-dependent phoneme recognition results are then compared to the context-independent phoneme recognition results.

Results

The isolated phoneme recognition accuracies, using triphones, are tabulated in Table D.7. In Fig. 6.7 the recognition accuracies for triphones are plotted with respect to the computational requirement as measured by the total number of parameters.

Figs. 6.8 and 6.9 compare the isolated phoneme recognition accuracies of different levels of context dependency when an increasing number of mixture DC GMMs and FC GMMs are used as the state distributions.

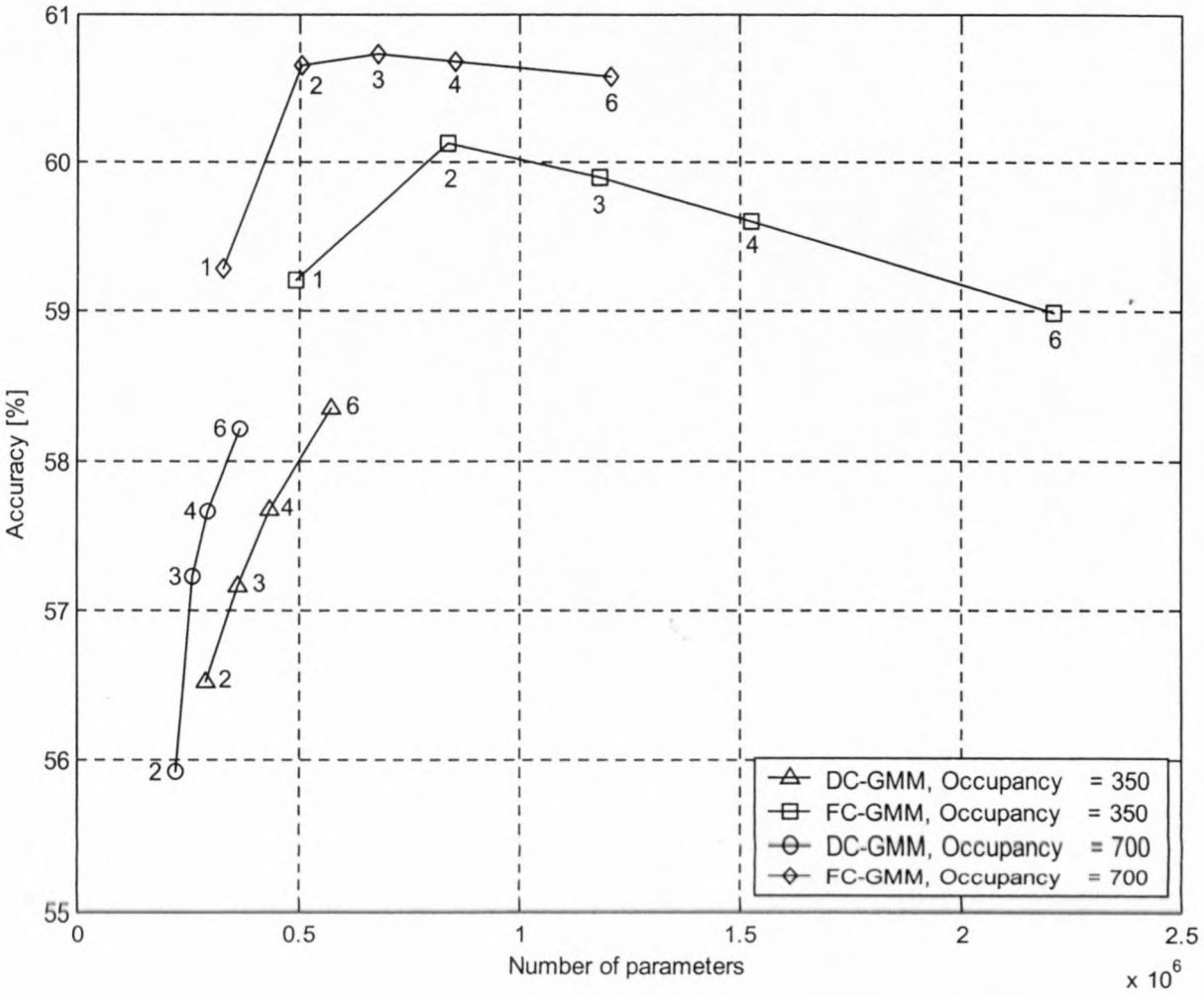

Figure 6.7: *Isolated phoneme recognition results on NTIMIT, using triphones to model context dependency.*

Interpretation

The triphone models with the least complex state distributions have a large computational requirement. It is interesting to note that the triphone models that were built by using a state occupancy of $\tau = 700$ perform better than the triphone models built by using a state occupancy of $\tau = 350$. This supports the theory that the biphone models did not outperform the monophone models due to a lack of training data. Furthermore, the recognition accuracy degrades when the number of FC GMM mixtures are increased past two mixtures, for $\tau = 350$, and three mixtures, for $\tau = 700$ respectively. The triphone models built by using a higher state occupancy require less computational time, because there are more triphones that are sharing the same state distributions. The best results, factoring both computational time and recognising accuracy, is two mixture FC GMMs with a state occupancy of $\tau = 700$. However, 24-mixture FC GMM monophones achieve nearly the same recognition accuracies, while having half the computational requirement.

Figs. 6.8 and 6.9 indicate that the use of context dependency does increase the recognition accuracy when the same type of state distribution is used. This is not an entirely fair comparison, as the number of parameters increases as the level of context dependency increases. The results also indicate that for isolated phoneme recognition left-context and right-context biphones model the phonemes equally well.

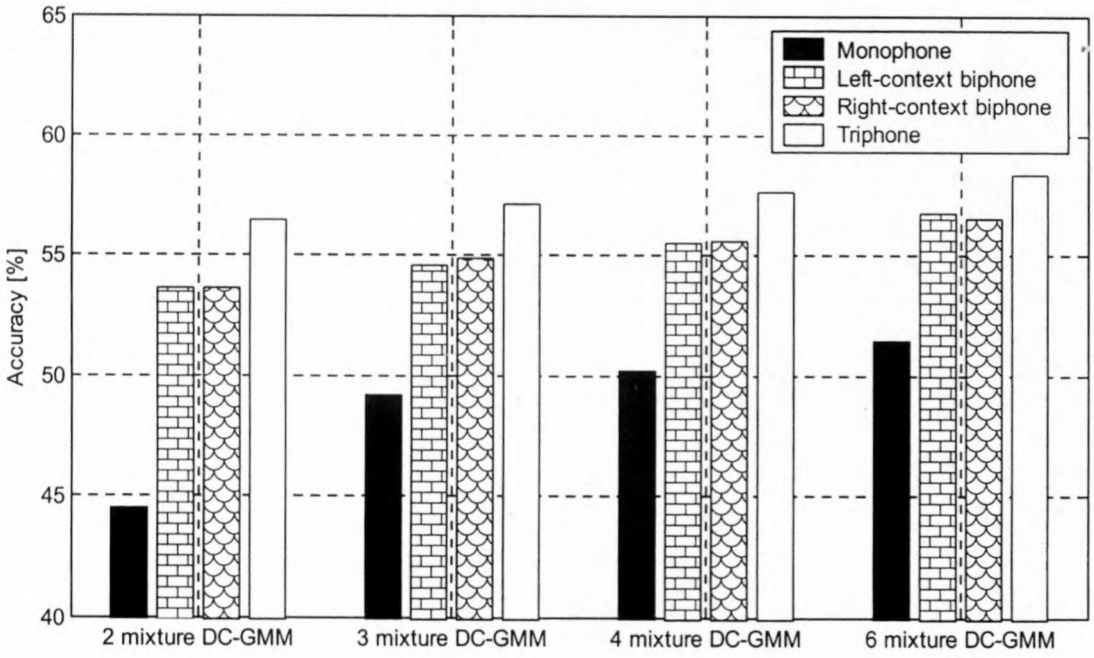

Figure 6.8: *Isolated phoneme recognition results on NTIMIT using increasing levels of context dependency and DC GMM state distributions.*

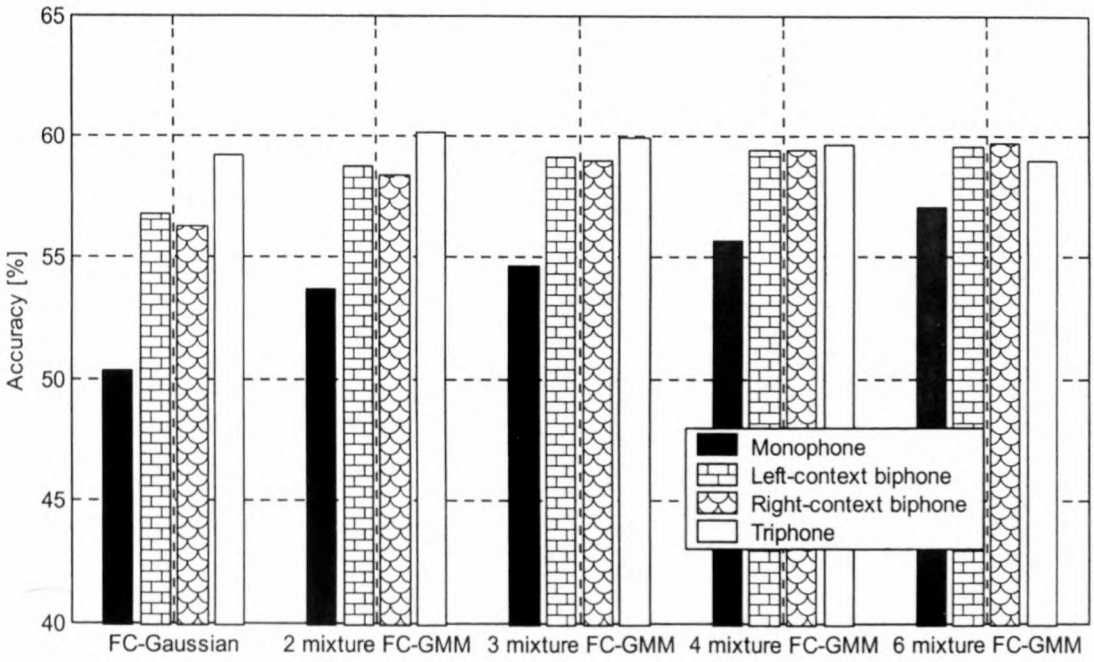

Figure 6.9: *Isolated phoneme recognition results on NTIMIT, using increasing levels of context dependency and FC GMM state distributions.*

6.3.4 Summary

The results of the context-dependent modelling experiments are summarised as follows:

- The use of parameter sharing is highly recommended as it vastly reduces the number of free parameters, while significantly increasing the isolated phoneme recognition accuracy.
- For a given type of state distribution, the use of context increases the isolated phoneme recognition accuracy. The deeper the level of context dependency, the higher the recognition accuracy.
- Left-context and right-context biphones perform equally well.
- The best context-dependent result is the same as the best context-independent result, while having a much higher computational requirement. This is probably due to a lack of enough training data, which is caused by using too small a state occupancy when performing the parameter sharing. This theory is supported by the isolated phoneme recognition results of triphones, in which a higher state occupancy leads to better recognition accuracy.

6.4 Continuous phoneme recognition: NTIMIT

6.4.1 Motivation

The previous experiments reported on the results for isolated phoneme recognition on the NTIMIT speech. However, speech rarely takes the form of isolated phonemes, and therefore it is necessary to perform continuous phoneme recognition experiments on the NTIMIT speech, in order to compile a baseline to which the continuous phoneme recognition experiments of the South African English database can be compared.

6.4.2 Experimental setup

The monophones, biphones and triphones constructed in the previous experiments are put into a context-dependent phoneme spotter. Therefore, all the context-dependent models are based on cross-word allophonic transcriptions. An example of a context-dependent phoneme spotter, using an alphabet of two monophones and their derived left-context biphones, is shown in Fig. 6.10. Informal experiments have shown that the transition probability between phonemes, a_{LINK}, dramatically influences the insertion and deletion rate, and therefore the recognition accuracy. When a language model is used during recognition, a_{LINK} is specified by the language model. But because no language model was used, it was necessary to choose a_{LINK}. It was found that the best results were achieved by choosing a_{LINK} to be inversely proportional to the number of context-dependent models in the phoneme spotter. Continuous phoneme recognition is performed by using increasing levels of context dependency in the context-dependent phoneme spotter. Four different levels of context dependency are used to model phoneme context, i.e. :

- Monophones
- Monophones and left- or right-context biphones
- Monophones and biphones
- Monophones, biphones and triphones

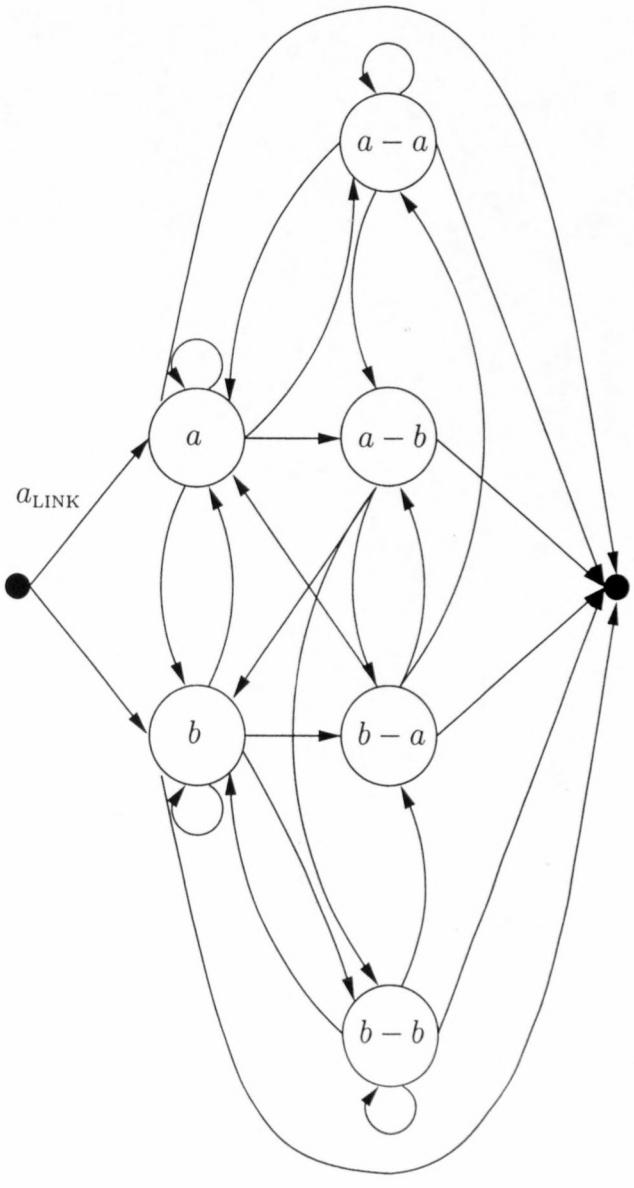

Figure 6.10: *An example of a two alphabet, context-dependent phoneme spotter.*

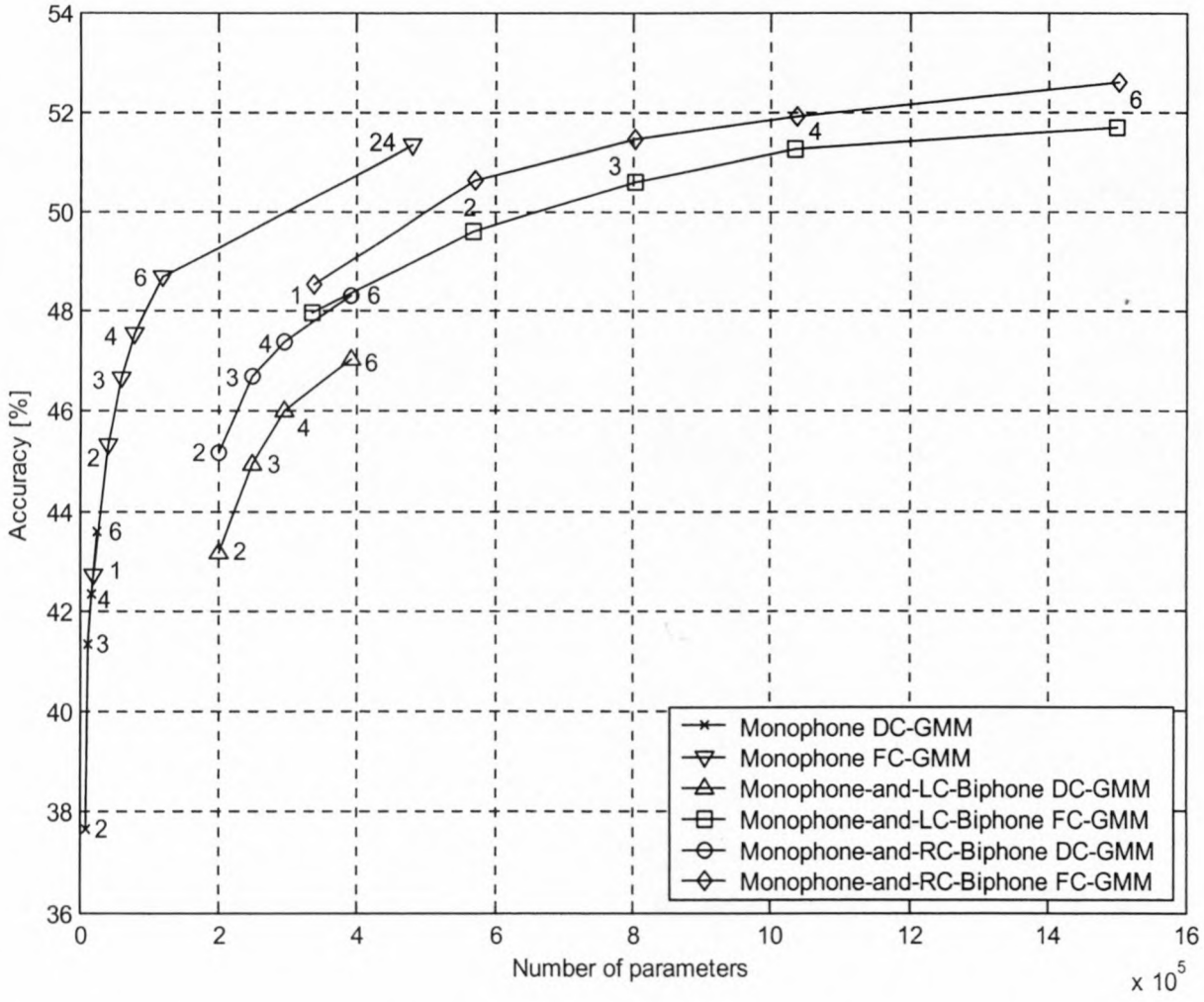

Figure 6.11: *Continuous phoneme recognition results on NTIMIT using monophones, monophones and left-context biphones, and monophones and right-context biphones to model context dependency.* 12^{th}*-order MFCCs are used as features.*

6.4.3 Results

For each of these levels of context dependency, continuous phoneme recognition experiments were performed and are tabulated in Tables D.8-D.12. Tables D.8-D.10 respectively tabulates the results for monophones, monophones and left-context biphones, and monophones and right-context biphones. Fig. 6.11 plots the continuous phoneme recognition accuracy with respect to the computational requirement, as measured by the total number of parameters, for monophone, monophone-and-LC-biphone, and monophone-and-RC-biphone recognition systems.

Fig. 6.12 plots the continuous phoneme recognition accuracy with respect to the computational requirement, as measured by the total number of parameters, for monophone-and-biphone and monophone-and-biphone-and-triphone based recognition systems. Figs. 6.13 and 6.14 illustrates the phoneme correct- and insertion rate, as well as the recog-

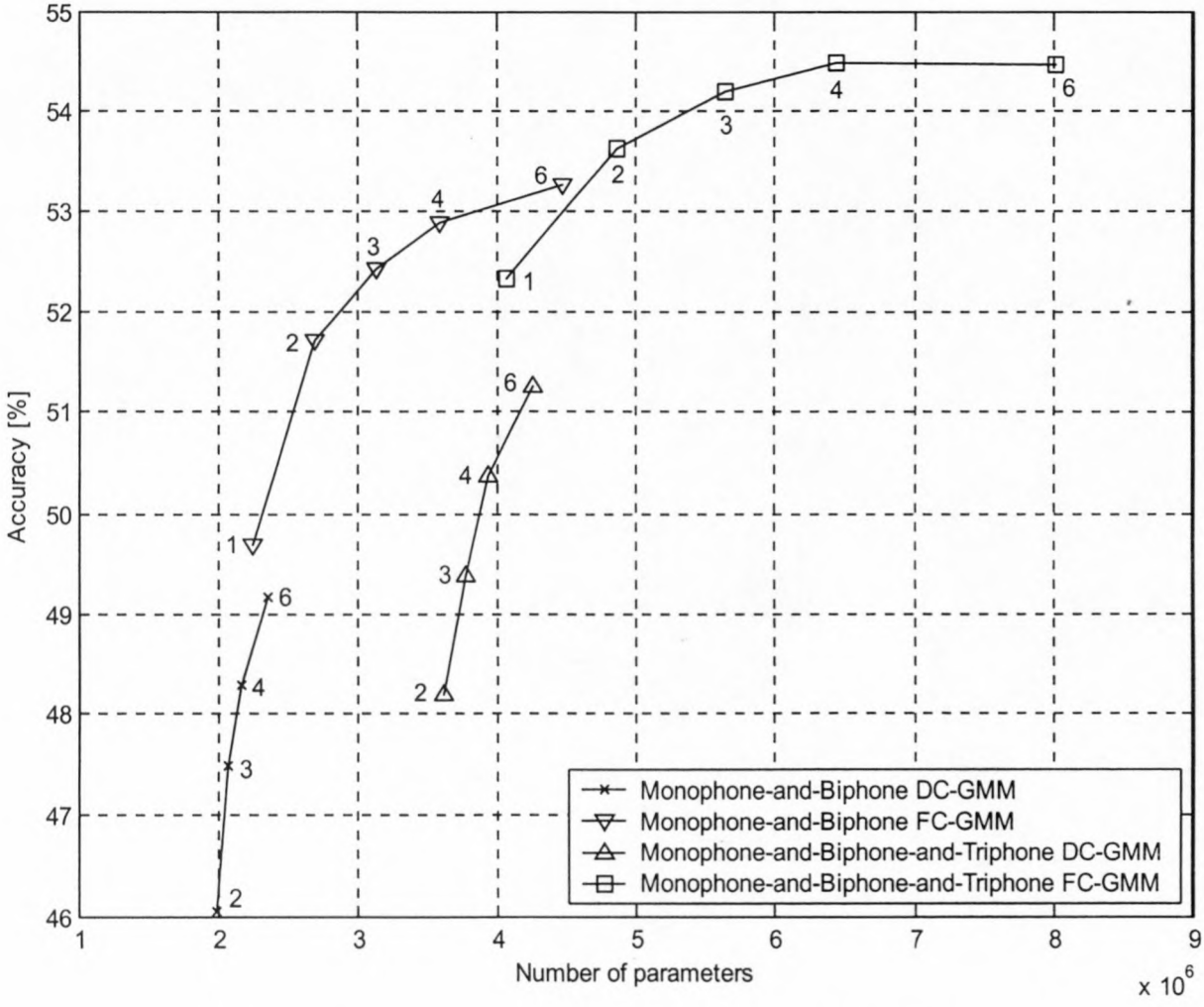

Figure 6.12: *Continuous phoneme recognition results on NTIMIT using mono- and biphones, and mono-, bi-, and triphones to model context dependency.* 12^{th}*-order MFCCs are used as features.*

nition accuracy of different levels of context dependency, for respectively DC GMM and FC GMM state distributions. The phoneme correct rate includes the substitution and deletion errors and represents the maximum recognition performance of a system if no extra phonemes were erroneously added.

6.4.4 Interpretation

Comparing the continuous and isolated phoneme recognition results, (Figs. 6.4 and 6.11 respectively), the graphs of recognition accuracy vs. computational requirement have the same general form for both the isolated and continuous phoneme recognition experiments. The difference is that the continuous phoneme recognition results are $\pm 10\%$ worse than the isolated phoneme recognition results, which is to be expected, as isolated phoneme recognition is an easier task than continuous phoneme recognition.

Figs. 6.13 and 6.14 indicate that if the same type of state output distribution is used, the use of context dramatically improves the recognition accuracy. This improvement comes at the cost of a large increase computational requirements. The relative improvement in accuracy from monophones to triphones is on average +20% with an associated increase in computational requirement by a factor of 66 (or more). What is interesting is that for all types of state distributions used, the monophone-and-RC-biphone system performed better than the monophone-and-LC-biphone system. This is consistent with the results of [26, 27]. Another interesting result is that the monophone spotters consistently have the lowest insertion error rate. A possible explanation for this is the relatively simple topology of the monophone spotters in comparison to the more complex context-dependent spotters. If the concept of perplexity is applied to the monophone spotters, the spotters would have a perplexity of 39 (any one of the 39 phonemes can follow a specific phoneme). The context-dependent spotters have a much higher perplexity, due to the modelling of context, and this could possibly cause a higher insertion rate.

The best context-independent recognition accuracy of 51.32% is achieved by using 24 mixture FC-GMM state distributions. The best context-dependent recognition accuracy of 54.47% is achieved by using mono-, bi-, and triphones modelled with four mixture FC GMM state distributions. This is only a relative improvement of +6.14%, while the context-dependent computational requirement is a factor of 13.4 times larger than the context-independent computational requirement. It is important to remember that the computational requirement recorded in this thesis is for phoneme recognition *without* the use of a language model. As a language model greatly decreases the number of legal phoneme sequences, the use of a language model will reduce the computational requirement of both context-dependent and context-independent modelling.

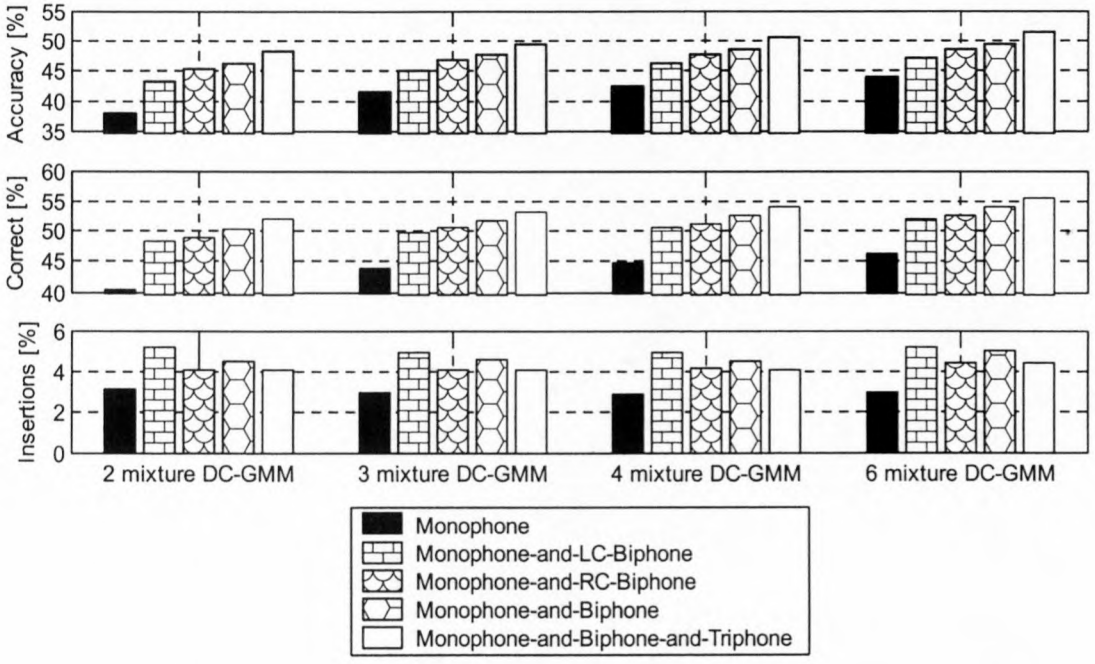

Figure 6.13: *Continuous phoneme recognition results on NTIMIT, using increasing levels of context dependency and DC GMM state distributions.*

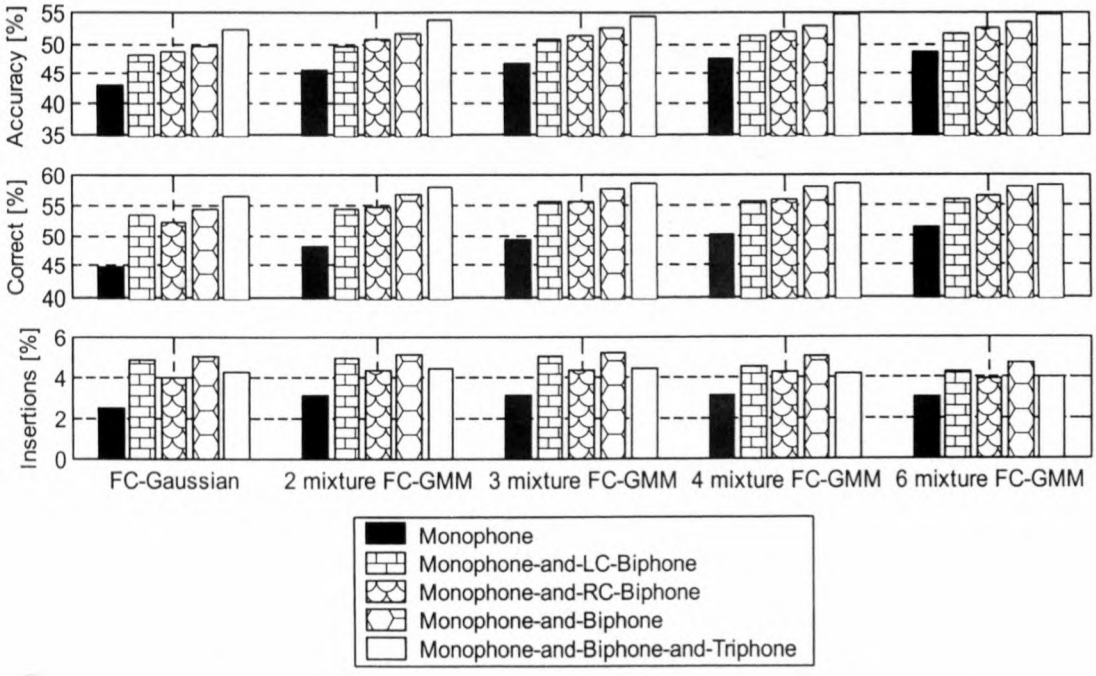

Figure 6.14: *Continuous phoneme recognition results on NTIMIT, using increasing levels of context dependency and FC GMM state distributions.*

6.5 Continuous phoneme recognition: S.A. English

All the experiments performed up to this point have been conducted to establish a baseline to which the continuous phoneme recognition experiments, performed on the untried South African English speech corpus, can be compared.

6.5.1 Motivation

It has been shown that context-independent and context-dependent phoneme models can be successfully trained on the NTIMIT speech corpus. The motivation of this experiment is to determine whether the phoneme modelling techniques can be applied directly to the modelling of South African English phonemes.

6.5.2 S.A. English Database

The African Speech Technology speech corpora consists of five different South African languages, which include English, Afrikaans, isiXhosa, isiZulu and Sesotho. The speech in each database consists of a mixture of read and spontaneous speech of mother-tongue speakers for a total of 38 to 40 utterances per speaker. The speech is recorded under normal telephone and cellphone conditions.

The English corpus (which is describe more fully in Appendix C.2) consists of five different dialects, which include Standard English, Black English, Coloured English, Asian English, Afrikaans English. Each of the different dialects represents a different ethnic group's version of English. For the purpose of this research, only the Standard English database is used, which consists of 287 unique speakers for a total of 11 433 utterances.

6.5.2.1 Format of speech data

The speech data is recorded by using a Dialogic analogue telephony card connected to the South African telephone network. The speech data recorded is stored in raw (headerless) format, containing audio samples at eight bits/sample, and sampled at 8 kHz, mono. Samples are encoded by using mu-law compression.

6.5.2.2 Data sets

When selecting the training- and testing sets of each database, care is taken to ensure that there are at least 100 occurences of each phoneme in the training set. If this first requirement is met, an attempt is made to include about 30 000 phonemes in the testing set, to make the results on the South African speech corpora comparable to the results on NTIMIT.

Training set

The training set consists of 9 275 utterances from 238 different speakers for a total of 159 560 phonemes and 87 292 non-speech events. This amounts to a total of 17.36 hours of data which consists of 1.62 hours of speech and 15.74 hours of non-speech events.

Testing set

The testing set consists of 1 813 utterances from different 49 speakers for a total of 30 515 phonemes and 11 072 non-speech events. This amounts to a total of 3.16 hours of data which consists of 0.70 hours of speech and 2.46 hours of non-speech events.

6.5.3 Experimental setup

The context-dependent, phoneme-based recogniser will be evaluated using 12^{th}-order MFCCs, which are post-processed using CMS, augmented with velocity and acceleration features, and dimensionally reduced by using LDA.

Sets of context-independent and context-dependent phonemes are constructed from the training data. According to [31] a word recognition system using word-internal context dependency is easier to implement than one using cross-word context dependency, because the total number of contexts is much smaller than in the cross-word case. Furthermore, with a word-internal-based recognition system every realisation of a word is the same and can be taken from a dictionary. The transcriptions of the South African English corpus are incomplete phonemic transcriptions and contain the word boundaries. Therefore, the South African English transcriptions are ideally suited to be modelled by using word-internal context dependency, because phonemic transcription implies that the transcription of each word is taken from a pronunciation dictionary. It was therefore decided that the South African corpus will be modelled by using word-internal, context-dependent phonemes. This means that only the phoneme context *within* words is taken into account when modelling context-dependent phonemes. The first and last phoneme of a word is modelled with context-independent phonemes or alternatively, is modelled with respectively right-context and left-context biphones. Decision tree-based state clustering is performed to share the state output pdf parameters across biphones derived from the same monophone. Because the total number of contexts is smaller, the problem of data scarcity is alleviated. All the allophone contexts in the training data set were "seen" contexts. It was decided to use the ideal case and only construct the models of unseen contexts that occured in the testing data set and not in the training data set. In other words, the complete set of phonemes consists of all the allophones that occur in the full speech corpus and no allophones are modelled that does not occur in the speech corpus. It must be stressed that no data from the testing set was used when modelling the allo-

phones, only data from the training set. Therefore, the allophones that occured in the testing set was used to determine *which* unseen allophone models to construct, not to determine the model parameters of the unseen allophone models. Therefore, the decision trees are constructed using only the allophones that occur in the training data set. Each of nine different non-speech events are modelled by using an eight-state, ergodic HMM, which uses a single full-covariance Gaussian state distribution.

It is not possible to perform isolated phoneme recognition experiments, because the available South African English database is only orthographically transcribed with phonemes determined using a pronunciation lexicon.[2] Therefore, a spotter-HMM is constructed with the set of phoneme HMMs, which is used to perform continuous phoneme recognition. No language model or grammar is assumed for the phoneme recognition experiments, but paths in the spotter is constrained so that only phoneme sequences are allowed that coincide with the context-dependent modelling. The insertion of extra incorrect non-speech event labels when the transcription indicates the correct phoneme is a non-speech event, is not added to the insertion error.

Continuous phoneme recognition is performed by using increasing levels of context dependency in the context-dependent phoneme spotter. Because the continuous phoneme experiments on the South African English requires approximately twice the time than that of the experiments on NTIMIT, and because the triphone and biphone phoneme spotters require too much memory during continuous recognition experiments, results for only the following levels of context dependency are available:

- Monophones
- Monophones and left- or right-context biphones

6.5.4 Results

For each of these levels of context dependency, continuous phoneme recognition experiments were performed and are tabulated in Tables D.13-D.15. The results are compared with the baseline results obtained on the NTIMIT speech corpus. Fig. 6.15 plots the continuous phoneme recognition accuracy with respect to the computational requirement, as measured by the total number of parameters, for monophone and monophone-and-biphone based recognition systems. Figs. 6.16 and 6.17 illustrates the phoneme correct- and insertion rate, as well as the recognition accuracy of different levels of context-depencency, for respectively DC GMM and FC GMM state distributions. The phoneme correct rate

[2]At the time of the experiments the South African English corpus was only orthographically transcribed. Subsequently, the corpus has been phonetically transcribed as well.

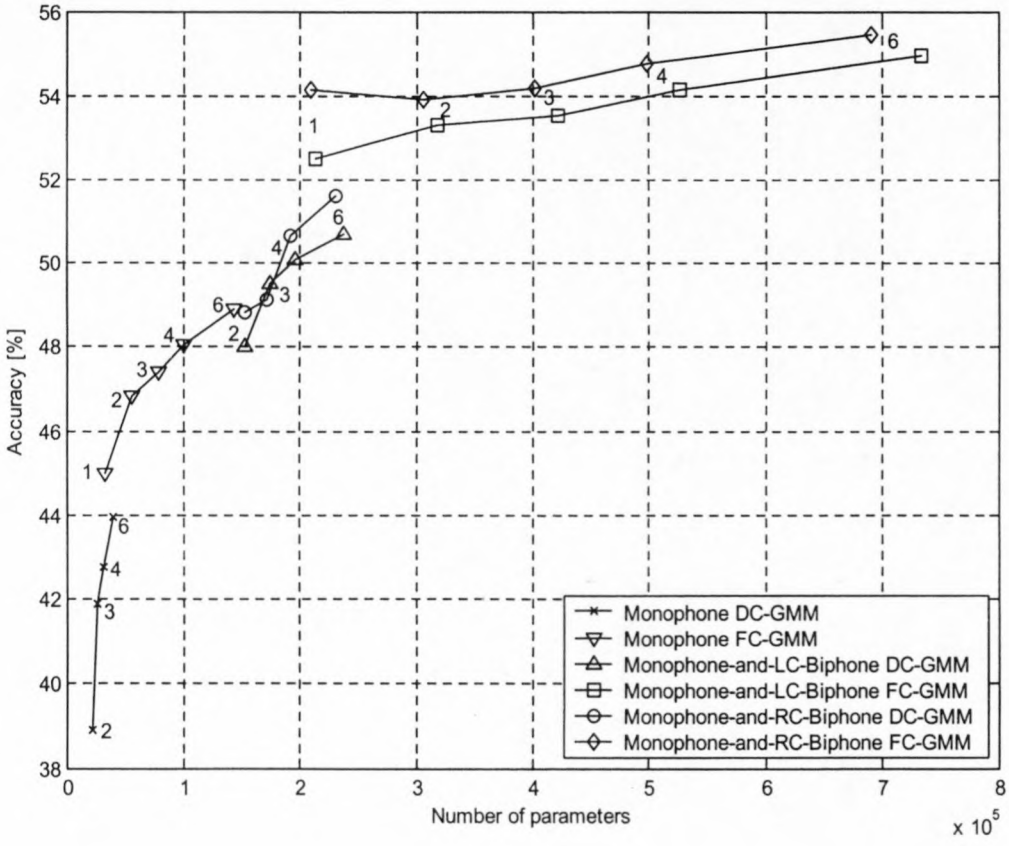

Figure 6.15: *Continuous phoneme recognition results on South African English, using monophones and biphones to model context dependency and* 12^{th}*-order MFCCs as feature vectors.*

includes the substitution and deletion errors and represents the maximum recognition performance of a system if no extra phonemes were erroneously added.

6.5.5 Interpretation

Comparing the continuous phoneme recognition results of the South African English to the results obtained on NTIMIT, it can be seen that the South African accuracies are higher. The recognition accuracies on the South African English are 2-6% better than the recognition accuracies on NTIMIT. Comparing Fig. 6.15 and 6.11, it can be seen that the results for monophones and biphones follow the same general pattern on both speech databases.

As in the case of the context-dependent, continuous phoneme recognition experiments performed on NTIMIT, the context-dependent results indicate that the inclusion of context increases the recognition accuracy, which is illustrated for DC GMM and FC GMM

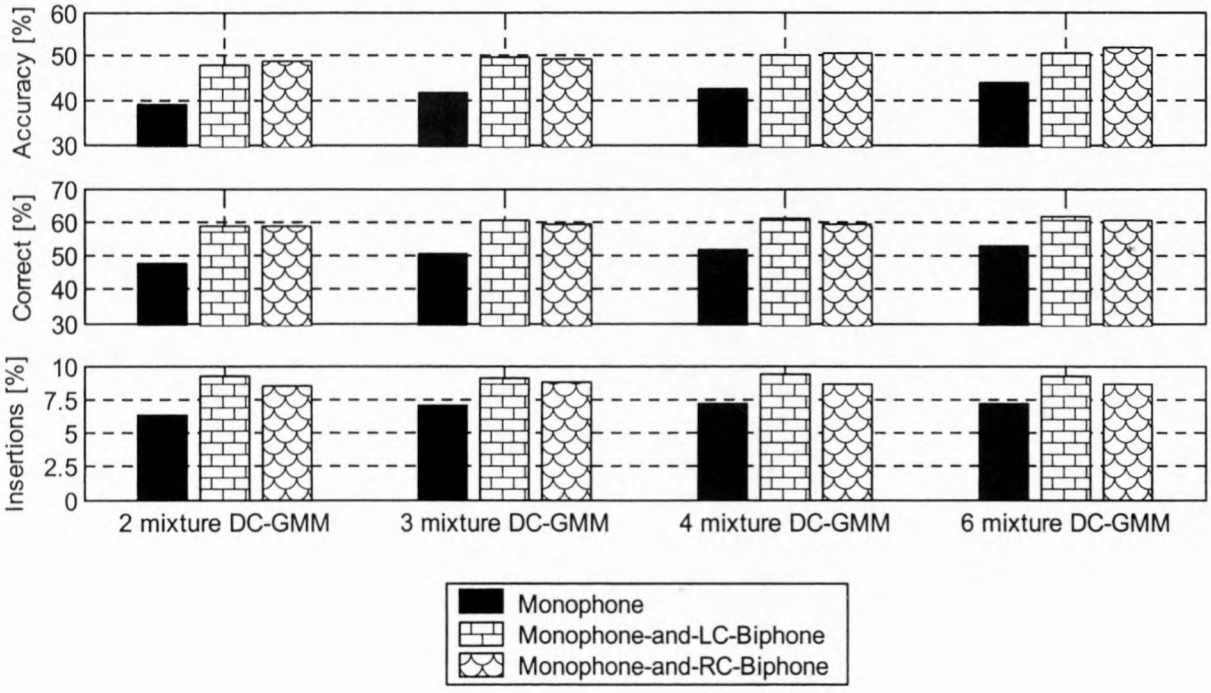

Figure 6.16: *Continuous phoneme recognition results on South African English, using increasing levels of context dependency and DC GMM state distributions.*

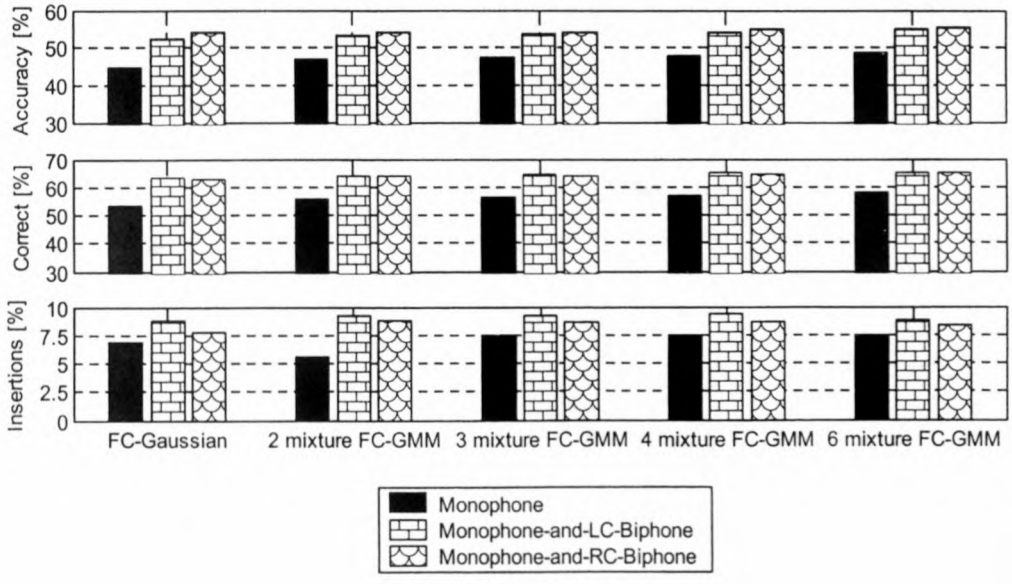

Figure 6.17: *Continuous phoneme recognition results on South African English, using increasing levels of context dependency and FC GMM state distributions.*

state distributions in Figs. 6.16 and 6.17. The insertion rates on the South African English is higher than for NTIMIT. This is to be expected as the NTIMIT phoneme models are trained on complete phonetic transcriptions and are therefore more accurate, while the South African English phoneme models are trained on incomplete phonemic transcriptions.

The results indicate that the untried South African English database is of a high enough quality that it can be used for the development of speech technology. It also means that the phoneme-modelling techniques, which have been proved to accurately model American English, can be directly applied to South African English. One would expect that the results obtained on NTIMIT would be better than the results obtained on South African English, but this does not seem to be the case. Possible reasons for the better results on South African English include:

- The use of cross-word context dependency on the NTIMIT database and word-internal context dependency on the South African English database. This causes the number of context-dependent phonemes to be much smaller for the South African English than for NTIMIT. Therefore, the South African English models suffer less from data scarcity than the NTIMIT phoneme models.

- Non-speech events are modelled with a single three-state, left-to-right HMM when performing experiments on the NTIMIT, while each of the nine different non-speech events are modelling with an eight-state ergodic HMM on the South African English database.

- In the NTIMIT database, the utterances which are spoken by all the speakers are not included in the train- or testset. In order to use all available data of the South African English databases, no effort was made to ensure that the same requirement was met for the experiments performed on the South African English databases.

6.6 Summary

This chapter reported on the results of the isolated and continuous phoneme recognition experiments performed on the South African English and NTIMIT speech corpora. Three different aspects of phoneme modelling was investigated by performing isolated phoneme recognition on the NTIMIT speech corpus. The three aspects were signal processing, statistical modelling of HMM state distributions and context-dependent modelling.

Based on the results of the signal processing experiments it was decided to used MFCCs as feature extraction technique for all subsequent experiments. The MFCCs are augmented with velocity and acceleration features and dimensionally reduced using LDA. Finally, cepstral mean subtraction is used to post-process the extracted features.

The statistical modelling experiments indicated that increasing the complexity of the state distributions results in an increase in the isolated recognition accuracy (within reason). The improvement in accuracy is not linearly related to the complexity, as the improvement saturates when the model becomes too complex to be reliably estimated from the limited amount of data. An interesting result was found when diagonal- and full-covariance GMM state distributions were compared on an equal-parameter basis. It was found that as the number of mixtures increase, full-covariance state distributions outperform diagonal-covariance distributions. This is interesting because modern LVCSR systems utilise diagonal-covariance distributions in an effort to reduce the computational requirements. A possible explanation is that there is not enough training data to reliably estimate the parameters of the large number of diagonal-covariance mixtures. Although DC GMM and FC GMM have an equal number of parameters, the FC GMM has a fewer number of mixtures and might therefore be able to more effectively utilise the training data.

The context-dependent modelling experiments show decision tree-based state clustering to be an effective technique of sharing the parameters of context-dependent phoneme models. The technique vastly reduces the number of free parameters, while significantly increasing the isolated phoneme recognition accuracy. It has also been shown that the use of context increases the isolated phoneme recognition accuracy. Generally, the deeper the level of context dependency, the higher the recognition accuracy (within reason).

Lastly, continuous phoneme recognition was performed using both context-dependent and context-independent phonemes. The levels of context that were investigated included monophones, left- and/or right context biphones, and triphones. As in the case of isolated phoneme recognition, it was shown that the use of context improves the continuous phoneme recognition accuracy. This increase in recognition accuracy is accompanied by a significant increase in computational requirement which is related to the level of context dependency of the phoneme models. Unexpectedly, the continuous recognition results obtained on the South African English was generally $2-6\%$ higher that the results obtained on NTIMIT. Slightly different experimental conditions was postulated as one possible reason for this result. However, the results still indicate that the untried South African English is of a high enough quality that it can be used for the development of speech technology.

Chapter 7

Conclusions

7.1 Concluding perspective

Two different approaches to phoneme modelling were examined in this thesis. The first approach was the modelling of phonemes without regard to context, while the other approach includes context when modelling phonemes. The signal processing and statistical modelling techniques examined relate to automatic speech recognition issues that influence both approaches in a similar manner and was done in an effort to choose the set of available techniques that maximises the recognition accuracy of a given approach. The biggest problem with the use of context when modelling phonemes, is a lack of sufficient training data. This problem, together with the problem of constructing context-dependent phonemes that do not occur in the training set, was elegantly solved by using decision tree-based state clustering, a parameter-sharing algorithm based on classification-and-regression trees.

It was expected that the inclusion of context when modelling phonemes would increase the performance of both isolated and continuous phoneme recognition. This did indeed occur, especially when the recognition accuracies were compared based on the same type of state distribution. However, the increase in performance came at the cost of a large increase in the computational requirement, as the shift from context-independent to context-dependent recognition gained a relative recognition improvement of +6% – at an increase in computation requirement by a factor of 66 or more. The large increase in computational requirement should not be given too much consideration. It must be remembered that a full Viterbi search was performed to obtain the most probable phoneme sequence for a given utterance. A full Viterbi search considers *all* paths through the more complex context-dependent spotter-HMM, which causes the large increase in computational requirement. However, the phoneme models was created for the purposes of LVCSR and most modern LVCSR systems use techiques such as fast match and path pruning techniques in order to keep the computational requirements to acceptable levels. Thus, the

question that still needs to be answered is how the modelling of context influences the computational requirement of a LVCSR recognition system. As mentioned in Chapter 1, building a LVCSR system and conducting phoneme-based continuous word-recognition is not a trivial exercise.

A comparison based on the same type of state distribution is not completely fair, because context-dependent phonemes in total have more state distributions than context-independent phonemes. Another comparison would be to compare the two approaches on an equal-parameter basis, in order to determine which approach is superior. However, it must be mentioned that modern speech recognisers do not perform continuous phoneme recognition, but perform continuous word recognition by building phoneme-based word models from pronunciation dictionaries. As the word models are built from pronunciation dictionaries, the comparison based on the same type of state distribution does provide some information on the expected continuous word recognition rate.

After the phoneme modelling techniques were examined by using the NTIMIT corpus, they were applied to the South African English corpus. It was found that the recognition results were similar to, or better, than the results obtained on the NTIMIT corpus, which indicates that the South African English corpus is of a high enough quality to be used in local HLT research and development.

7.2 Comparison to prior work

The results of the feature extraction experiments is supported by the research of Psutka *et al.* [34], which indicates that the recognition accuracies of both MFCCs and PLPs are comparable, but that PLP provides slightly better performance, and that using 12 cepstral coefficients is better than using 18 coefficients. Psutka *et al.* performed telephone-based, speaker-independent, continuous speech recognition of Czech, using a zero-gram language model. The speech corpus that was used consisted of 100 speakers reading 40 sentences over the telephone, indicating that the conditions of the experiments are very similar to the conditions of the feature extraction experiments. Although they performed continuous word recognition, while this thesis is concerned with phoneme recognition, the results are still comparable as the word recognition engine is phoneme-based. When using the MFCC parameterisation, they report a continuous recognition accuracy of 82.6% for 12^{th}-order MFCCs (and 22 filter banks) and a recognition accuracy of only 81.1% for 18^{th}-order MFCCs. For the PLP parameterisation, they report a recognition accuracy of 83.3% for 12^{th}-order PLPs and do not report any results for 18^{th}-order PLPs. However, their best recognition accuracy of 83.9% for PLPs is achieved when using 8^{th}-order PLPs, and their results indicate that the recognition accuracy decreases when using more cepstral coefficients.

There are currently very few publications on speaker-independent phoneme recognition (especially phoneme recognition without the aid of language models), as the current research focus is generally on speaker-independent continuous word recognition, utilising language models. However, a number of studies concerned with multilingual speech recognition [42, 43, 44] and language identification [46] on the GlobalPhone corpus [41, 47] were published by the Interactive Systems Laboratory (ISL) of Carnegie Mellon University. In these studies there are references to continuous phoneme recognition of telephone speech. In all the publications the speech recognition engine is based on 3 000 triphone HMMs aided with a language model. Each HMM state is modelled with 32 mixture Gaussian pdfs (it is unclear what type of covariance is used for the Gaussian pdfs) and the parameters of these pdfs are shared using decision tree-based state clustering. The signal processing techniques used are 13^{th}-order MFCCs augmented with velocity and acceleration features, and power and zero crossing rate. The features are post-processed by using CMS and LDA, which results in 32-dimensional features. For their English monolingual recognition engine, a phoneme error rate (PER) of 46.4% is reported for 46 phonemes and a trigram language model perplexity of 9.2 ([42, p. 1], [43, p. 86], [44, p. 38]), which is equivalent to a continuous phoneme recognition accuracy of 53.6%. These results compare favourably to the continuous recognition accuracy of 54.47% that was achieved on the NTIMIT corpus, using four mixture FC-Gaussian triphones and the accuracy of 55.34% that was achieved on the South African English corpus using six mixture FC-Gaussian right-context biphones, because no language model was used to obtain the results of this thesis. If a language model is included during recognition, it can be expected that the recognition accuracies will increase.

7.3 Outstanding issues

There are a number of refinements to phoneme recognition that was not addressed by this thesis, as well as a number of issues that was uncovered by the results. The outstanding issues include:

- The use of grammar and a language model during continuous phoneme recognition should be investigated. The purpose of such experiments would be to determine whether the computational requirement of context-dependent models can be appreciably reduced.

- Continuous word recognition experiments should be performed by using the same phoneme models to determine the extent to which phoneme-recognition accuracy give rise to word-recognition accuracy.

- One of the biggest stumbling blocks with the present implementation of the pho-

neme-recognition system is the large computational requirements. For obvious reasons the recognition time must be as small as possible, but the long delay between implementing a technique and obtaining the experimental results tends to cause errors in the experiments as well. It is therefore necessary to build a faster decoder. Possible techniques include multi-pass and fast-match recognition techniques.

- Biphone models that utilises a higher state occupancy when performing decision tree-based state clustering should be constructed to determine whether saturation of recognition accuracy is due to lack of training data.

- Decision tree-based state clustering does not implement merging of clusters or complex splitting questions. Research shows that these two aspects of tree construction improves the parameter sharing, and the recognition performance of the context-dependent models [31].

- In this thesis a VQ-based method was used to increase the number of mixtures of the context-dependent models. This requires an extra iteration through the training data. Young, Odell and Woodland uses another method of increasing the GMM mixture number, called mixture-splitting, which is not dependent on data [31, 50, 51, 52]. Mixture-splitting should be implemented and compared to the VQ-based method to determine whether mixture splitting is a better method of constructing context-dependent GMMs than the VQ-based method used in this thesis.

- When the recognition system processes segments containing no speech, the phoneme recognition results are skewed by incorrectly recognised phonemes inside the non-speech segments. Speech detection pre-processing should be performed to eliminate the influence of non-speech segments on the recognition accuracy.

- This research applies phoneme-modelling only to the English dialect of the South African English speech database. The techniques should be applied to the other four dialects of South African English (i.e. Asian, Black, Cape and Afrikaans English). Interesting topics of reseach include the influence of different dialects on each other, as well as the performance of multilingual phoneme models (models trained simultaneously on all the dialects).

- The phoneme-modelling techniques can be applied to the isiXhosa database in order to determine whether the techniques, which successfully model South African English, deliver acceptable results on isiXhosa. This topic of research includes the possibility of examining multilingual phoneme models by using both isiXhosa and South African English.

Bibliography

[1] Bahl, L.R., *et al.*, "Context Dependent Modeling of Phones in Continuous Speech Using Decision Trees." *Proceedings of the DARPA Speech and Natural Language Processing Workshop*, 1991, pp. 264–268.

[2] Baker, R., "Continuous speech word recognition via centisecond acoustic states." in *Proc. ASA Meeting* (Washington, DC), April 1976. Referenced in [37].

[3] Barnard, C., Botha, E.C., "Analogue Telephone Speech Recorder: User's Manual." tech. rep., African Speech Technology, September 2000.

[4] Baum, L.E., Eagon, J.A., "An Inequality with Applications to Statistical Estimation for Probabilistic Functions of Markov Processos and to a Model for Ecology." *Bulletin of American Mathematical Society*, 1967, Vol. 73, pp. 360–363.

[5] Bengio, Y., "Markovian Models for Sequential Data." *Neural Computing Surveys*, 1999, Vol. 2, pp. 129–169.

[6] Bishop, Christopher M., *Neural Networks for Pattern Recognition*. New York: Oxford University Press Inc., 1995.

[7] Botha, E.C., Louw, P.H., Roux, J.C., van den Heuvel, M., "Corpus Design, Speaker Recruitment Documentation and Standards for Telephone Speech Database." tech. rep., African Speech Technology, March 2001.

[8] De Wet, F., "Isolated Word Speech Recognition in Xhosa." Master's thesis, University of Pretoria, 1999.

[9] Deller, John R., Jr., *et al.*, *Discrete-Time Processing of Speech Signals*. Upper Saddle River, New Jersey: Prentice-Hall, 1993.

[10] Devijver, D.A., Kittler, J., *Pattern Recognition, a statistical approach*. Englewood Cliffs, New Jersey: Prentice-Hall International, 1982.

[11] Digalakis, V.V., Monaco, P., Murveit, H., "Genones: Generalised Mixture Tying in Continuous Hidden Markov Model-Based Speech Recognisers." *IEEE Trans. of Speech and Audio Processing*, July 1996, Vol. 4, No. 4, pp. 281–293.

[12] Du Preez, J.A., *Efficient High-Order Hidden Markov Modelling*. PhD thesis, University of Stellenbosch, 1997.

[13] Ferguson, F.D., "Hidden Markov Models: An Introduction." in *Hidden Markov Models for Speech*, (Princeton, NJ), 1980.

[14] Garofolo, J.S., Lamel, L.F., Fisher, W.M., Fiscus, G.J., Pallett, D.S. and Dahlgren, N.L., "The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CDROM." Printed documentation, 1992.

[15] Graybill, F.A., *Matrices with Applications in Statistics*. second edition. Belmont, California: Wadsworth Publishing Company, Inc., 1983.

[16] Hermansky, H., "Perceptual Linear Predictive (PLP) analysis of speech." *Journal of the Acoustic Society of America*, 1990, Vol. 87, No. 4, No. 4, pp. 1738–1752.

[17] Hon, H-W., Lee, K-F., "Recent Progress in Robust Vocabularyu-Independent Speech Recognition." in *Proc. DARPA Speech and Natural Language Processing Workshop*, (Pacific Grove), 1991. Referenced in [31].

[18] Huang, X., Acero, A., Hon, H.W., *Spoken language processing: A guide to theory, algorithm and system development*. Upper Saddle River, New Jersey: Prentice-Hall, 2001.

[19] Hwang, M-Y., Huang, X., "Shared-Distribution Hidden Markov Models for Speech Recognition." *IEEE Transaction on Speech and Audio Processing*, October 1993, Vol. 1, No. 4, pp. 414–420.

[20] Hwang, M.Y., Huang, X., Alleva, F.A., "Predicting Unseen Triphones with Senones." *IEEE Transactions on Speech and Audio Processing*, November 1996, Vol. 4, No. 6, pp. 412–419.

[21] Jankowski, C., "The NTIMIT Speech Database." Documentation on NTIMIT CD-ROM, January 1991.

[22] Jankowski, C., Kalyanswamy, A., Basson, S., Spitz, J., "NTIMIT: A Phonetically Balanced, Continuous Speech, Telephone Bandwidth Speech Database." *Proc. of ICASSP*, April 1990.

[23] Jelinek, F., *Statistical Methods for Speech Recognition*. Massachusetts: The Massachusetts Institute of Technology Press, 1997.

[24] Jurafsky, D., Martin, J.M., *Speech and Language Processing - An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Upper Saddle River, New Jersey 07458: Prentice Hall, 2000.

[25] Kittler, J., Young, P.C., "A new approach to feature selection based on the Karhunen-Loeve expansion." *Pattern Recognition*, 1973, Vol. 5, pp. 335–352.

[26] Lee, K-F., *Automatic Speech Recognition — The Development of the SPHINX System*. Boston: Kluwer Academic Publishers, 1989.

[27] Lee, K-F., "Speaker-independent phone recognition using hidden Markov models." *IEEE Trans. on Acoustics, Speech, and Signal Processing*, November 1989, Vol. 37, No. 11, pp. 1641–1648.

[28] Levinson, S.E., "Continuously Variable Duration Hidden Markov Models for Automatic Speech Recognition." *Computer Speech and Language*, 1986, pp. 29–45.

[29] Markel, J.D., Gray Jr., A.H., *Linear Prediction of Speech*. Berlin, N.Y.: Springer Verlag, 1976. Referenced in [49].

[30] Ming, J., O'Boyle, P., Owens, M., Smith, F.J., "A Bayesian Approach for Building Triphone Models for Continuous Speech Recognition." *IEEE Transaction on Speech and Audio Processing*, November 1999, Vol. 7, No. 6, pp. 678–684.

[31] Odell, J.J., *The Use of Context in Large Vocabulary Speech Recognition*. PhD thesis, University of Cambridge, 1995.

[32] Pallett, D.S., Fiscus, J.G., Fisher, W.M., Garafolo, J.S., Lund, B.A., Przybocki, M.A., "1994 Benchmark Tests for the ARPA Spoken Language Program." in *Proc. ARPA Workshop on Human Language Technology*, (Merrill Lynch Conference Centre), pp. 49–74, 1994. Referenced in [31].

[33] Proakis, J.G., Manolakis, D.G., *Digital Signal Processing: Principles, Algorithms, and Applications*. second edition. Inglewood Cliffs, New Jersey.: Macmillan Publishing Company, 1992. Referenced in [49].

[34] Psutka, J., Müller, L., Psutka, J.V., "Comparison of MFCC and PLP parameterizations in the Speaker Independent Continuous Speech Recognition Task." *Eurospeech*, 2001.

[35] Rabiner, L.R., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition." *Proceedings of the IEEE*, February 1989, Vol. 77, No. 2, pp. 257–286.

[36] Rabiner, L.R., Juang, B.H., "An Introduction to Hidden Markov Models." *IEEE ASSP Magazine*, January 1986, Vol. 1, No. 3, pp. 4–16.

[37] Rabiner, L.R., Juang, B.H., *Fundamentals of Speech Recognition*. Englewood Cliffs, New Jersey: Prentice-Hall, 1993.

[38] Rabiner, L.R., Schafer, R.W., *Digital Procesing of Speech Signals*. Englewood Cliffs, N.J.: Prentice Hall, 1978.

[39] Reichl, W., Chou, W., "Robust Decision Tree State Tying for Continuous Speech Recognition." *IEEE Trans. of Speech and Audio Processing*, September 2000, Vol. 8, No. 5, pp. 555–566.

[40] Russell, M.J., Moore, R.K., "Explicit Modelling of State Occupancy in Hidden Markov Models for Automatic Speech Recognition." *Int. Conf. on Acoustics, Speech and Signal Processing*, 1985, pp. 5–8.

[41] Schultz, T., "GlobalPhone: A Multilingual Speech and Text Database developed at Karlsruhe University." *Proc. of the Int. Conf. of Spoken Language Processing*, September 2002.

[42] Schultz, T., Waibel, A., "Language adaptive LVCSR through Polyphone Decision Tree Specialization." *Workshop on Multi-lingual Interoperability in Speech Technology*, September 1999, pp. 85–90.

[43] Schultz, T., Waibel, A., "Polyphone Decision Tree Specialization for Language Adaptation." *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, June 2000.

[44] Schultz, T., Waibel, A., "Language-independent and language adaptive acoustic modelling for speech recognition." *Speech Communication*, 2001, Vol. 35, pp. 31–51.

[45] Schultz, T., Waibel, A., "Language-independent and language adaptive acoustic modelling for speech recognition." *Proc. of the Int. Conf. of Spoken Language Processing*, September 2002.

[46] Schultz, T., Westphal, M., Waibel, A., "LVCSR-based Language Identification." *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, May 1996.

[47] Schultz, T., Westphal, M., Waibel, A., "The GlobalPhone Project: Multilingual LVCSR with JANUS-3." *Multilingual Information Retrieval Dialogs: 2nd SQEL Workshop*, April 1997.

[48] "The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus." Documentation on CD-ROM "NIST Speech Disk CD1-1.1", December 1990.

[49] Van der Merwe, R, "Variations on Statistical Phoneme Recognition - A Hybrid Approach -." Master's thesis, University of Stellenbosch, 1997.

[50] Young, S.J., "The General Use of Tying in Phoneme-Based HMM Speech Recognisers." in *Proc. ICASSP*, (San Francisco), pp. 569–572, 1992. Referenced in [31].

[51] Young, S.J., Odell, J.J., Woodland, P.C., "Tree-Based State Tying for High Accuracy Acoustic Modelling." in *Proc. of the ARPA Workshop on Human Language Technology*, (Merril Lynch Conference Centre), pp. 286–291, 1994.

[52] Young, S.J., Woodland, P.C., "State clustering in hidden Markov model-based continuous speech recognition." *Computer Speech and Language*, 1994, pp. 369–383.

[53] Zhao, Y., "A Speaker-Independent Continuous Speech Recognition System Using Continuous Mixture Gaussian Density HMM of Phoneme-Sized Units." *IEEE Trans. of Speech and Audio Processing*, July 1993, Vol. 1, No. 3, pp. 345–361.

Appendix A

Feature extraction techniques

The following discussion on the feature extraction techniques is a combination of the literature found in [9, 18, 23, 37, 49].

A.1 Linear predictive cepstral coefficients – LPCC

The basic assumption of linear prediction is that the speech signal $s(n)$ can be represented by a linear combination of the past values and the input value $e(n)$

$$s(n) = \sum_{i=1}^{p} a_i s(n-i) + \Theta_0 u(n) \tag{A.1}$$

where Θ_0 is the gain factor and p is the order of the linear predictor. Because the input signal $u(n)$ is unknown, the speech signal can only be approximated as $\tilde{s}(n)$ where

$$\tilde{s}(n) = \sum_{i=1}^{p} a_i s(n-i) \tag{A.2}$$

The prediction error between each pair of samples $e(n)$ is defined as

$$e(n) = s(n) - \tilde{s}(n) \tag{A.3}$$

The LP parameters $\hat{a}_i$ is determined by minimising E,

$$\begin{aligned} E &= \sum_{n} e^2(n) \\ &= \sum_{n} \left[s(n) - \sum_{i=1}^{p} a_i s(n-i) \right]^2, \end{aligned} \tag{A.4}$$

the total mean square error with respect to the LP parameters a_i. Ideally, the error summation should be done over the whole signal $s(n)$, but in practice only a small number of samples are available for the estimation of the LP parameters, since the stationarity

assumption only holds true for short speech segments. Therefore, the error summations are performed over N samples. Define the short-term speech and error segments as

$$\begin{aligned} s_n(m) &= s(n+m) \\ e_n(m) &= e(n+m) \end{aligned} \qquad 0 \leq m \leq N-1 \tag{A.5}$$

then the total mean squared error at time n becomes

$$E_n = \sum_{m=0}^{N-1} e_n^2(m) \tag{A.6}$$

The total mean square error is quadratic in the LP parameters and therefore has a unique global minimum that can be determined by setting the partial derivatives with respect to the LP parameters equal to zero, i.e.

$$\frac{\partial E_n}{\partial a_j} = 0, \quad 1 \leq j \leq p \tag{A.7}$$

The partial derivative with respect to a_j is

$$\begin{aligned} \frac{\partial E_n}{\partial a_j} &= \sum_{m=0}^{N-1} \left[2 \left\{ s_n(m) - \sum_{i=1}^{p} a_i s_n(m-i) \right\} \cdot \{-s_n(m-j)\} \right] \\ &= -2 \sum_{m=0}^{N-1} s_n(m) s_n(m-j) + 2 \sum_{i=1}^{p} a_i \sum_{m=0}^{N-1} s_n(m-i) s_n(m-j) \end{aligned} \tag{A.8}$$

The minimisation leads to the p *normal equations*, which is given in terms of the estimates $\hat{a}_i$ of the LP parameters a_i:

$$\sum_{i=1}^{p} \hat{a}_i \sum_{m=0}^{N-1} s_n(m-i) s_n(m-j) = \sum_{m=0}^{N-1} s_n(m) s_n(m-j), \quad j = 1, 2, \ldots, p \tag{A.9}$$

Define the short-term covariance $\phi_{i,j}$ of $s_n(m)$ as

$$\phi_{i,j} = \sum_{m=0}^{N-1} s_n(m-i) s_n(m-j) \tag{A.10}$$

then the *normal equations* can be compactly written as

$$\phi_{i,0} = \sum_{j=1}^{p} \hat{a}_j \cdot \phi_{i,j} \quad i = 1, 2, \ldots, p \tag{A.11}$$

or in matrix form:

$$\begin{bmatrix} \phi_{1,0} \\ \phi_{2,0} \\ \vdots \\ \phi_{p,0} \end{bmatrix} = \begin{bmatrix} \phi_{1,1} & \phi_{1,2} & \cdots & \phi_{1,p} \\ \phi_{2,1} & \phi_{2,2} & \cdots & \phi_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{p,1} & \phi_{p,2} & \cdots & \phi_{p,p} \end{bmatrix} \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \vdots \\ \hat{a}_p \end{bmatrix} \tag{A.12}$$

The two methods of solving the Eq. A.12 differs in the way that the speech segment $s_n(m)$ is treated outside the analysis frame $0 \leq m \leq N-1$. Only the *autocorrelation* method will be discussed.

A.1.1 Autocorrelation method

The autocorrelation method assumes that the speech segment $s_n(m)$ is zero outside the interval $0 \leq m \leq N-1$ and thus the speech segment becomes a weighted signal, i.e.

$$s_n(m) = \begin{cases} s(n+m) \cdot w(m), & 0 \leq m \leq N-1 \\ 0, & \text{otherwise} \end{cases} \tag{A.13}$$

where $w(m)$ is a finite length window, which is zero outside the interval $0 \leq m \leq N-1$. Based on using the weighed signal of Eq. A.13, the total squared error becomes

$$E_n = \sum_{m=0}^{N-1+p} e_n^2(m) \tag{A.14}$$

and the short-term covariance of the speech segment can be expressed as

$$\begin{aligned} \phi_{i,j} &= \sum_{m=0}^{N-1+p} s_n(m-i)s_n(m-j), \quad \begin{matrix} 1 \leq i \leq p \\ 0 \leq j \leq p \end{matrix} \\ &= \sum_{m=0}^{N-1-(i-j)} s_n(m)s_n(m+i-j), \quad \begin{matrix} 1 \leq i \leq p \\ 0 \leq j \leq p \end{matrix} \\ &= r_{i-j} \end{aligned} \tag{A.15}$$

where r_{i-j} is the unbiased *autocorrelation estimate* for a finite data window. As the autocorrelation function is symmetric, the *normal equations* can be expressed in terms of the autocorrelation:

$$r_i = \sum_{j=1}^{p} r_{|i-j|}\hat{a}_j, \quad 1 \leq i \leq p \tag{A.16}$$

or in matrix form:

$$\begin{aligned} \mathbf{r}_s &= \mathbf{R}_s \cdot \hat{\mathbf{a}} \\ \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_p \end{bmatrix} &= \begin{bmatrix} r_0 & r_1 & \cdots & r_{p-1} \\ r_1 & r_0 & \cdots & r_{p-2} \\ \vdots & \vdots & \ddots & \vdots \\ r_{p-1} & r_{p-2} & \cdots & r_0 \end{bmatrix} \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \vdots \\ \hat{a}_p \end{bmatrix} \end{aligned} \tag{A.17}$$

and has the solution

$$\hat{\mathbf{a}} = \mathbf{R}_s^{-1}\mathbf{r}_s, \tag{A.18}$$

where $\mathbf{R}$ is in the highly symmetrical Toeplitz form. Levinson-Durbin recursion is used to recursively calculate the order-(i) solution from the order-$(i-1)$ solution. The algorithm is summarised as follows:

1. **Initialisation**:

$$E^{(0)} = r_0 \tag{A.19}$$

2. **Recursion**: Cycle through $(i) = 1, 2, 3, \ldots, p$

$$\begin{aligned} k_i &= \frac{r_i - \sum_{j=1}^{i-1} \alpha_j^{(i-1)} r_{i-j}}{E^{(i-1)}} && \text{(A.20)} \\ E^{(i)} &= (1 - k_i^2) E^{(i-1)} && \text{(A.21)} \\ \alpha_i^{(i)} &= k_i && \text{(A.22)} \\ \alpha_j^{(i)} &= \alpha_j^{(i-1)} + k_i \alpha_{i-j}^{(i-1)}, \quad j = 1, 2, \ldots, i-1 && \text{(A.23)} \\ & && \text{(A.24)} \end{aligned}$$

3. **Termination**: Estimated order-p LP-parameters are:

$$\hat{\mathbf{a}} = \{\alpha_1^{(p)}, \alpha_2^{(p)} \ldots, \alpha_p^{(p)}\} \tag{A.25}$$

The coefficients k_m are known as the reflection coefficients and $E^{(i)}$ is the total mean squared prediction error of the order-i LP filter.

A.1.2 LPC to cepstrum conversion

If the speech signal $s(n)$ is produced by a minimum-phase, all-pole, model as depicted in Fig. 2.5, the log magnitude spectrum is given by

$$\log |S(\omega)|^2 = \log \frac{\hat{\Theta}^2}{|A(z)|^2} \tag{A.26}$$

By using Eqs. A.1 and A.26, the following recursive relationship between the LP parameters $\hat{\mathbf{a}}$ and the cepstral coefficients $c(m)$ [29] exists:

$$\begin{aligned} c(0) &= \log \hat{\Theta} && \text{(A.27)} \\ c(m) &= \hat{a}_m + \sum_{k=1}^{m-1} \left(\frac{k}{m}\right) \hat{a}_{m-k} c(k), \quad m = 1, 2, \ldots, p && \text{(A.28)} \\ c(m) &= \sum_{k=m-p}^{m-1} \left(\frac{k}{m}\right) \hat{a}_{m-k} c(k), \quad m > p && \text{(A.29)} \end{aligned}$$

This cepstrum, derived from the minimum-phase, all-pole magnitude spectrum, is known as the *LPC cepstrum* in contrast with the real cepstrum calculated from the Fourier spectrum.

A.2 Mel-frequency cepstral coefficients – MFCC

The MFCC is a representation defined as the real cepstrum of a windowed short-time signal derived from the FFT of that signal. The difference from the real cepstrum is that a non-linear frequency scale is used, which approximates the behaviour of the auditory system. The short-time Fourier spectrum of the speech signal $s(n)$ can either be estimated by the short-time DFT [33] or by using a filter bank approach. Deller *et al.* [9] gives the short-time DFT of a windowed speech signal as

$$S_n(k) = \sum_{m=0}^{N-1} s_n(m)w(m)e^{-jk(\frac{2\pi}{k})m}, \quad k = 0, 1, \ldots, N-1 \tag{A.30}$$

Since the spectrum of the speech frame is convoluted in the frequency domain by the spectrum of the window function $w(m)$, the window must be chosen to limit the amount of spectral leakage which occurs. A *Hamming* window is typically used. For the filter bank approach, a set of bandpass filters is used to break the speech signal up into the relevant subbands in the frequency domain. The centre frequencies and bandwidths of these filters are chosen in such a way that they cover the complete frequency range of interest. Define a filterbank with Q filters ($q = 1, 2, \ldots, Q$), where filter q is a triangular filter given by:

$$H_q(k) = \begin{cases} 0 & k < f(q-1) \\ \frac{k-f(q-1)}{f(q)-f(q-1)} & f(q-1) \leq k \leq f(q) \\ \frac{k(q+1)-k}{f(q+1)-f(q)} & f(q) \leq k \leq k(q+1) \\ 0 & k > f(q+1) \end{cases} \tag{A.31}$$

Define f_l and f_h as the lowest and highest frequencies of the filterbank in Hz, Fs the sampling frequency in Hz, Q the number of filters, and N the size of the FFT. The boundary points $f(q)$ are uniformly spaced on the *mel-scale*:

$$f(m) = \left(\frac{N}{F_s}\right) B^{-1}\left(B(f_l) + q\frac{B(f_h) - B(f_l)}{Q+1}\right), \tag{A.32}$$

where the mel-scale B is given by

$$B(f) = 1125 \log\left(1 + \frac{f}{700}\right) \tag{A.33}$$

and its inverse

$$B^{-1}(b) = 700(e^{(b/1125)} - 1). \tag{A.34}$$

The mel-frequency scale and the triangular filterbank used to compute MFCCs is respectively shown in Fig. A.1 and Fig. A.2. The log-energy $S(q)$ is computed at the output of

each filter as

$$P(q) = \log \left[\sum_{k=1}^{N-1} |S_n(k)|^2 H_q(k) \right], \quad 0 \leq q < Q \tag{A.35}$$

Therefore, mel-frequency cepstral coefficients (MFCCs) is the discrete cosine transform of the Q filter outputs:

$$c(m) = \sum_{q=0}^{Q-1} P(q) \cos \left(\frac{\pi m (q + 0.5)}{Q} \right) \tag{A.36}$$

A.3 Perceptual linear prediction – PLP

Perceptual Linear Prediction (PLP) [16] uses the standard Durbin recursion (Eqs. A.19 to A.25) to compute LP parameters, which are converted to LPC cepstrum, using the conversion of Section A.1.2. Unlike LPCC, PLP does not compute the autocorrelation coefficients in the time domain using Eq. A.15, but replaces the magnitude spectrum $|S(\omega)|^2$ with a perceptually motivated power spectrum to obtain a parameterisation of speech that is more consistent with human hearing. The perceptually-motivated power spectrum incorporates three concepts from the psychoacoustics of hearing:

- **Critical-bank spectral resolution**: a critical band defines a frequency range in psychoacoustic experiments. When two competing tones fall within the critical band, the one with the higher power masks the other to human hearing. However, both tones are perceived when the two tones are separated in frequency by more

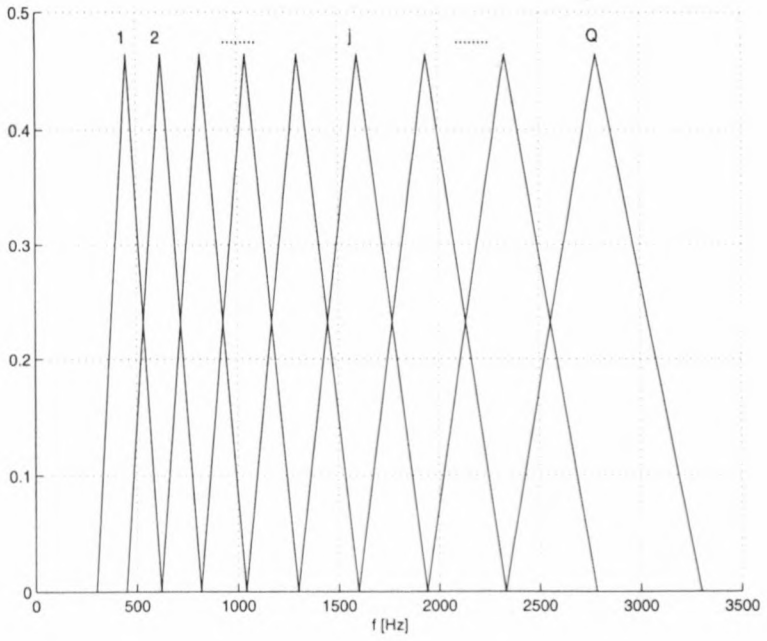

Figure A.1: *Triangular filter bank uniformly spaced on the mel-frequency scale.*

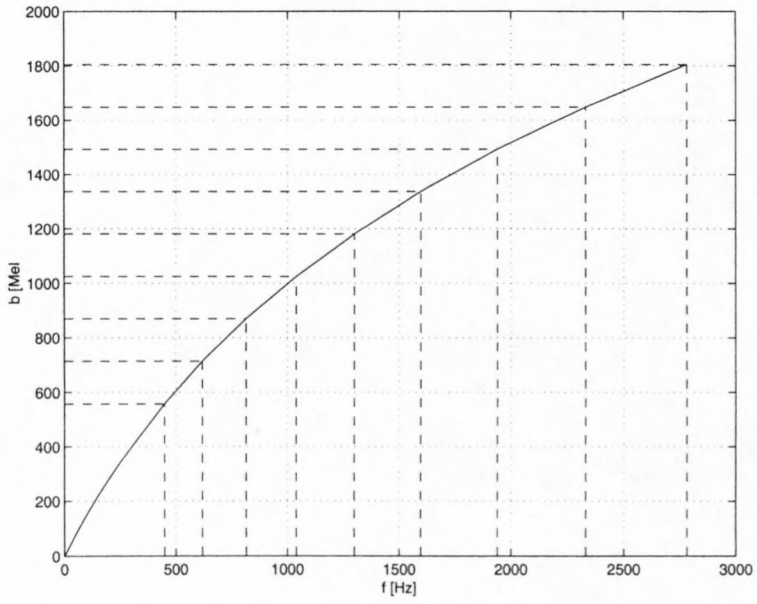

Figure A.2: *Filterbank centre frequencies on mel-frequency scale.*

than the width of the critical bank. The *Bark* scale (which is similar to the mel-frequency scale) relates acoustic frequency to perceptual frequency resolution so that one Bark covers one critical band. The Bark scale is defined as

$$\Omega(\omega) = 6 \log \left(\frac{\omega}{1200\pi} + \sqrt{\left(\frac{\omega}{1200\pi}\right)^2 + 1} \right) \tag{A.37}$$

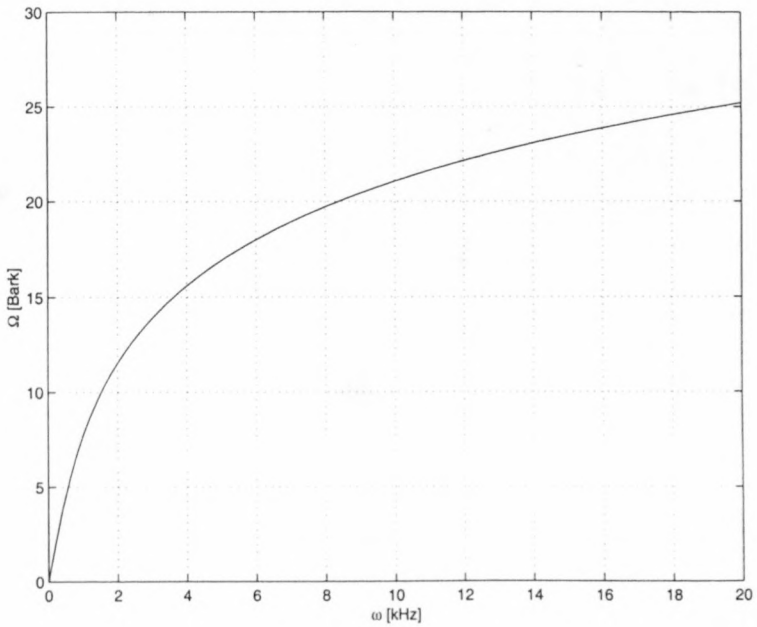

Figure A.3: *The Bark scale.*

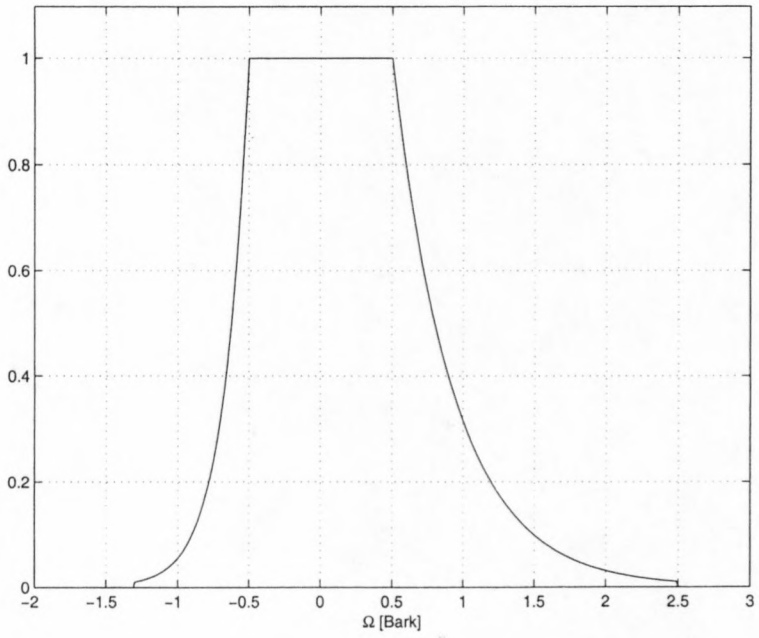

Figure A.4: *Critical-band filter transfer function.*

and is shown in Fig. A.3. The limited perceptual frequency resolution, as expressed by the notion of critical bands, is imposed on the high-resolution magnitude spec-

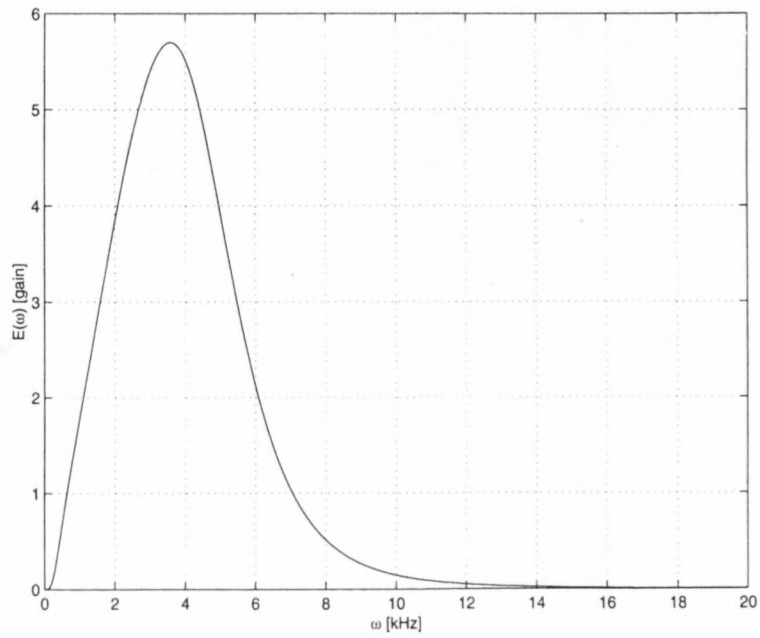

Figure A.5: *Equal-loudness preemphasis filter transfer function.*

trum of the speech by convolution with a critical bank filter function defined as

$$\Phi(\Omega) = \begin{cases} 0 & \Omega < -1.3 \\ 10^{2.5(\Omega+0.5)} & -1.3 \leq \Omega < -0.5 \\ 1 & -0.5 \leq \Omega < 0.5 \\ 10^{-(\Omega-0.5)} & 0.5 \leq \Omega < 2.5 \\ 0 & \Omega > 2.5 \end{cases} \tag{A.38}$$

and shown in Fig. A.4

- **Equal-loudness preemphasis**: human hearing is the most sensitive to frequencies around in the range of $3.5kHz$. The purpose of equal-loudness preemphasis is to compensate for the non-uniform sensitivity of human hearing to different frequencies. The equal-loudness preemphasis filter takes the form of

$$E(\omega) = \frac{(\omega^2 + 56.8 \cdot 10^6)\omega^4}{(\omega^2 + 6.3 \cdot 10^6)^2(\omega^2 + 0.38 \cdot 10^9)(\omega^6 + 9.58 \cdot 10^{26})}, \tag{A.39}$$

which is shown in Fig. A.5.

- **Intensity-loudness power law**: There is a non-linear relationship between the intensity of a sound and its perceived loudness by the human ear. The purpose of the intensity-loudness filter is to simulate this non-linear, intensity-loudness relationship. The intensity-loudness power law is shown as Fig. A.6 and expressed as

$$L(\omega) = \sqrt[3]{I(\omega)}, \tag{A.40}$$

where $L(\omega)$ is the perceived loudness and $I(\omega)$ is the intensity of the sound.

The PLP procedure can be summarised as follows:

1. Window the time-domain speech signal $s(n)$.
2. Compute the short-term power spectrum $P(\omega)$.
3. Map the short-term power spectrum to the bark scale by using Eq. A.37
4. Reduce the spectral resolution in such a way so that it resembles the ear by convolution with the critical-band filter (Eq. A.38).
5. Downsample to typically 18 spectral samples (approximately one sample every one Bark).

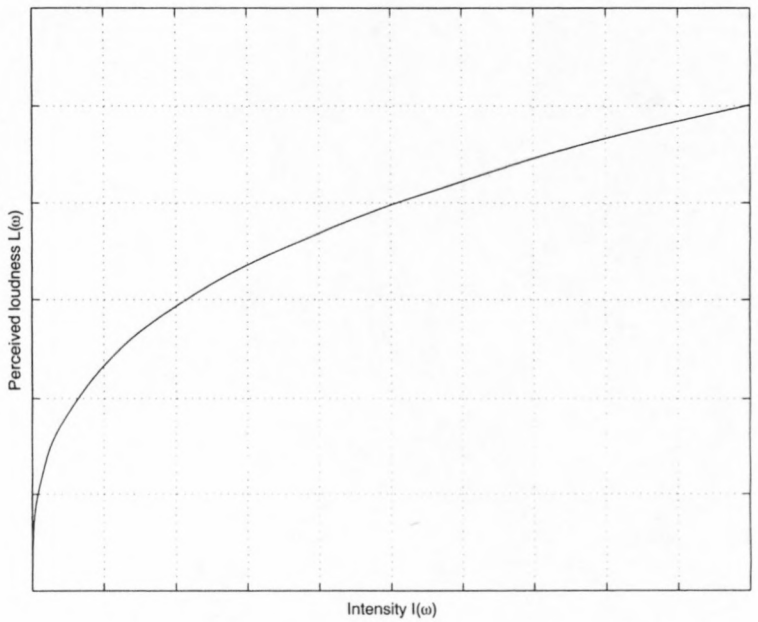

Figure A.6: *Intensity-loudness power law.*

6. Perform equal-loudness preemphasis to compensate for the non-uniform sensitivity of the human ear to different frequencies (Eq. A.39)

7. Simulate non-linear, intensity-loudness power law (Eq. A.40).

8. Perform LP analysis on the perceptually motivated power spectrum to obtain the PLP parameters (Section A.1).

9. Convert the PLP parameters to cepstral coefficients to obtain PLP cepstral coefficients (PLPs) (Section A.1.2).

Appendix B

Decision tree question sets

The questions used to construct the decision trees were questions obtained from the doctoral thesis of J.J. Odell [31]. Because question set was based on the LIMSI phoneme set it was necessary to translate it to the NTIMIT and new phoneme set. The question sets are divided into three main categories, i.e. *vowel-*, *consonant* and *general* questions. The sets are respectively listed, in IPA format, in Tables B.1, B.2 and B.3. Simple position independent question regarding the allophone context were used. An allophone $\mathcal{C}_\rho$ derived from the base phone ρ is written in the form "LC-ρ+RC" where LC and RC respectively denotes the left- and right phonetic context of the phone ρ. The general question concerning whether the left context of ρ is a NASAL or not, is generated as follows:

$\{$ m-ρ, m̥-ρ, n-ρ, ɲ-ρ, ŋ-$\rho\}$

The general question concerning whether the right context of ρ is a NASAL or not, is generated as follows:

$\{\rho$+m, ρ+m̥, ρ+n, ρ+ɲ, ρ+ŋ $\}$

The complete set of questions $\mathbb{Q}$ is compiled by generating, for each base phone, all general, vowel, and consonant questions concerning the left- and right phonetic context. The decision-tree questions incorporate linguistic knowledge into the clustering procedure by ensuring that unseen contexts are grouped together with those contexts which one would expect to be linguistically similar [31]. The clustering procedure automatically chooses the most important questions to use. In order to allow each phonetic context to be modelled seperately when enough data is available, a question was included for each phone. These questions are not shown since each question just has a single phone as member.

Table B.1: *Vowel questions.*

FRONT VOWEL	i, iː, y, yː, e, eː, ɛ, ɛː, æ, æː
CENTRAL VOWEL	ɪ, ø, øː, œ, œː, ʌ, ə, a, aː
BACK VOWEL	u, uː, ʊ, o, oː, ɔ, ɔː, ɑː, ɒ
LONG	ɔ, ɔː, iː, yː, uː, eː, øː, oː, ɛː, œː, ɜ, ɜː, aː, ɑː, æː, ø, e͡ɪ, ɔ͡ɪ, ɪ͡ə, ɛ͡ə ʊ͡ə, u͡a, ə͡i, i͡u, i͡a, œ͡y, u͡i, ə͡u, a͡i, o͡i
SHORT	i, y, ɪ, u, ʊ, e, o, ɛ, œ, ʌ, a, ɒ, ə, æ, a͡ʊ, a͡u, ə͡ʊ, ʌ͡ɪ, a͡ɪ, ɛ͡ɪ
DIPHTONG	a͡ʊ, a͡u, ə͡ʊ, ʌ͡ɪ, a͡ɪ, ɛ͡ɪ, e͡ɪ, ɔ͡ɪ, ɪ͡ə, ɛ͡ə, ʊ͡ə, u͡a, ə͡i, i͡u, i͡a, œ͡y, u͡i, ə͡u a͡i, o͡i
FRONT START	ɛ͡ɪ, e͡ɪ, ɛ͡ə, i͡u, i͡a
FRONTING	ʌ͡ɪ, a͡ɪ, ɔ͡ɪ, ə͡i, u͡i, a͡i, o͡i
HIGH	i, iː, y, yː, ɪ, u, uː, ʊ, e, eː, ø, øː, o, oː, a͡ʊ, a͡u, ə͡i, i͡u, u͡i, ə͡u, a͡i o͡i
MEDIUM	ɛ, ɛː, œ, œː, ɜː, ə, ɔ, ɔː, ə͡ʊ, ʌ͡ɪ, a͡ɪ, ɛ͡ɪ, e͡ɪ, ɔ͡ɪ, ɪ͡ə, ɛ͡ə, ʊ͡ə, œ͡y
LOW	ʌ, a, aː, ɑː, ɒ, æ, æː, i͡a
ROUNDED	y, yː, u, uː, ʊ, o, oː, ɔ, ɔː, ɒ, a͡ʊ, a͡u, ə͡ʊ, ɔ͡ɪ, ʊ͡ə, u͡a, œ͡y, u͡i, ə͡u, o͡i
UNROUNDED	i, iː, e, eː, ø, øː, ɛ, ɛː, œ, œː, ɜː, ʌ, a, aː, ɑː, ə, æ, æː, ʌ͡ɪ, a͡ɪ, ɛ͡ɪ e͡ɪ, ɪ͡ə, ɛ͡ə, ʊ͡ə, ə͡i, i͡u, i͡a, a͡i
REDUCED	ʌ, i
IVOWEL	i, iː, y, yː, ɪ
EVOWEL	e, eː, ø, øː, ɛ, ɛː
AVOWEL	a, aː, ɑː
OVOWEL	o, oː, ɔ, ɔː, ɒ, æ, æː
UVOWEL	u, uː, ʊ, œ, œː, ʌ, ɜː, ə

The use of the questions is demonstrated as follows:
The allophone cluster of left-context biphones, derived from monophone b, consisting of {/a-b/, /ʌ-b/, /ə-b/, /p-b/, /k-b/}, is split with the general question VOWEL. This is done by examining the left context (*) of all the biphones in the allophone cluster i.e. /*-b/. All the biphones with left context (*) that form part of the group phonemes of the general question VOWEL are placed in the one allophone cluster. The rest are placed in the other allophone cluster. The two new allophone clusters would therefore be {/a-b/, /ʌ-b/, /ə-b/ } and { /p-b/, /k-b/ }.

Table B.2: *Consonant questions.*

UNVOICED	p, pʰ, p', p͡ʃ', p͡ʃʰ, t, tʰ, t', t͡ɬ, t͡ʃ', t͡sʰ, t͡s', c', cʰ, k, kʰ, k' ʔ, ɾ, f, θ, s, ʃ, h, ɦ
VOICED	b, b̥, b, b͡ʒ, t͡ɬ, d, d̥, ɟ, k͡x', g, m, m̥, n, ɲ, ŋ, r, ʀ, v, ð z, ʒ, x, ɬ, ɮ, ɹ, j, l, ɓ, w, d͡ʒ, d͡ɮ, d͡z
FRONT CONSONANT	p, pʰ, p', b, b̥, m, m̥, ɾ, f, v, w, ɓ
CENTRAL CONSONANT	t, tʰ, t', t͡ɬ, t͡s', t͡sʰ, d, d̥, d͡ʒ, d͡ɮ, d͡z, ʔ, n, ɲ, ŋ, r, ʀ, ʒ k͡x', x, ɬ, ɮ, ɹ, l, s, z, θ, ð
BACK CONSONANT	p͡ʃ', p͡ʃʰ, t͡ʃ, t͡ʃʰ, c', cʰ, k, kʰ, k', g, h, ɦ, j
FORTIS	t͡ʃ, f, k, kʰ, k', p, pʰ, p', p͡ʃ', p͡ʃʰ, s, ʃ, t, tʰ, t', θ, ɾ, t͡ʃ, t͡ʃʰ t͡s', t͡sʰ, c', eʰ, ɟ, ʔ
LENIS	b, b̥, b͡ʒ, ɓ, d, d̥, d͡z, ð, g, d͡ʒ, v, z, ʒ
NON-FORTIS LENIS	l, m, m̥, n, ɲ, ŋ, r, ʀ, x, h, ɦ, ɬ, ɮ, ɹ, j, l, w, d͡ɮ, t͡ɬ, k͡x'
CORONAL	p͡ʃ', p͡ʃʰ, b͡ʒ, t, tʰ, t', t͡ɬ, t͡ʃ, t͡ʃʰ, t͡s', t͡sʰ, d, d̥, d͡ʒ, d͡ɮ, d͡z c', cʰ, ɟ, k͡x', n, ɲ, r, ʀ, ɾ, θ, ð, s, ʃ, z, ʒ, x, ɬ, ɮ, ɹ, l
NON-CORONAL	p, pʰ, p', b, b̥, ɓ, k, kʰ, k', g, ʔ, m, m̥, f, v, ŋ, h, ɦ, j, w
ANTERIOR	p, pʰ, p', b, b̥, t, tʰ, t', t͡ɬ, t͡s', t͡sʰ, d, d̥, d͡ɮ, d͡z, m, m̥ n, ɲ, ɾ, v, θ, ð, s, z, ɬ, ɮ, l, ɓ, w
NON-ANTERIOR	p͡ʃ', p͡ʃʰ, b͡ʒ, t͡ʃ, t͡ʃʰ, d͡ʒ, c', cʰ, ɟ, k, kʰ, k', k͡x', g, ʔ, ŋ, r, ʀ ʃ, ʒ, x, h, ɦ, ɹ, j
CONTINUENT	p, pʰ, p', p͡ʃ', p͡ʃʰ, b͡ʒ, t͡ɬ, t͡s', t͡sʰ, d͡ɮ, d͡z, m, m̥, n, ɲ, ŋ, r ʀ, f, v, θ, ð, s, z, ʃ, ʒ, x, ɦ, h, ɬ, ɮ, ɹ, j, l, w
NON-CONTINUENT	b, b̥, ɓ, t, tʰ, t', t͡ʃ, d͡ʒ, t͡ʃʰ, c', cʰ, d, d̥, ɟ, k, kʰ, k', k͡x' g, ʔ, ɾ
POSITIVE STRIDENT	p͡ʃ', p͡ʃʰ, b͡ʒ, t͡ʃ, t͡ʃʰ, t͡s', t͡sʰ, d͡ʒ, c', cʰ, ɟ, s, z, ʃ, ʒ
NEGATIVE STRIDENT	f, v, θ, ð, h, ɦ, ɓ
NEUTRAL STRIDENT	p, pʰ, p', b, b̥, t, t', t͡ɬ, d, d̥, d͡ɮ, d͡z, k, kʰ, k', k͡x', g ʔ, m, m̥, n, ɲ, ŋ, r, ʀ, x, ɬ, ɮ, ɹ, j, l, w
SYLLABIC	m̥
VOICED STOP	b, b̥, ɓ, d, g
UNVOICED STOP	p, pʰ, p', k, kʰ, k', t, tʰ, t', ʔ
FRONT STOP	b, b̥, ɓ, p, pʰ, p'
CENTRAL STOP	d, t, tʰ, t', ʔ
BACK STOP	g, k, kʰ, k'

VOICED FRICATIVE	ɟ, d͡ʒ, ð, v, z, ʒ
UNVOICED FRICATIVE	t͡ʃ, c', cʰ, f, s, ʃ, θ
FRONT FRICATIVE	f, v
CENTRAL FRICATIVE	ð, s, θ, z
BACK FRICATIVE	t͡ʃ, t͡ʃʰ, d͡ʒ, ʃ, ʒ
AFFRICATE	p͡ʃ', p͡ʃʰ, b͡ʒ, t͡ɬ, t͡ʃ, t͡ʃʰ, t͡s', t͡sʰ, d͡ʒ, d͡ʝ, d͡z, k͡x'
NON-AFFRICATE	ð, θ, f, s, ʃ, v, z, ʒ

Table B.3: *General questions.*

STOP	b, b̥, ɓ, d, d̥, c', cʰ, ɟ, k, kʰ, k', g, ʔ, p, pʰ, p', t, tʰ, t', ɾ
NASAL	m, m̥, n, ɲ, ŋ
FRICATIVE	θ, ð, d͡ʒ, d͡z, b͡ʒ, x, k͡x', p͡ʃ', p͡ʃʰ, t͡ʃ, t͡ʃʰ, t͡s', t͡sʰ, ɟ, f, v, s, z, ʃ, ʒ
LIQUID	d͡ʝ, t͡ɬ, r, ʀ, h, ɦ, ɬ, ʝ, ɹ, l, w, j
VOWEL	i, iː, y, yː, ɪ, u, uː, ʊ, e, eː, ø, øː, o, oː, ɛ, ɛː, œ, œː, ɜː, ʌ, ɔ, ɔː, a, aː ɑː, ɒ, ə, æ, æː, a͡ʊ, a͡u, ə͡ʊ, ʌ͡ɪ, a͡ɪ, ɛ͡ɪ, e͡ɪ, ɔ͡ɪ, ɪ͡ə, ɛ͡ə, ʊ͡ə, u͡a, ə͡i, i͡u, i͡a, œ͡y u͡i, ə͡u, a͡i, o͡i
FRONT	i, iː, y, yː, e, eː, ɛ, ɛː, æ, æː, p, pʰ, p', b, b̥, m, m̥, ɾ, f, v, w, ɓ
CENTRAL	ɪ, ø, øː, œ, œː, ʌ, ə, a, aː, b͡ʒ, t, tʰ, t', t͡ɬ, t͡s', t͡sʰ, d, d̥, d͡ʒ, d͡ʝ, d͡z, ʔ n, ɲ, ŋ, r, ʀ, ʒ, k͡x', x, ɬ, ʝ, ɹ, l, s, z, θ, ð
BACK	u, uː, ʊ, o, oː, ɔ, ɔː, ɑː, ɒ, p͡ʃ', p͡ʃʰ, t͡ʃ, t͡ʃʰ, c', cʰ, k, kʰ, k', g, h, ɦ, j
CLICKS	ǀ, ǀ̃, ǀ̰, ǀ̤, ǀʰ, ǁ, ǁ̃, ǁ̰, ǁ̤, ǁʰ, !, !̃, !̰, !̤, !ʰ

Appendix C

Speech corpora

C.1 *DARPA* NTIMIT speech corpus

The information regarding the NTIMIT speech corpus has been obtained from [14, 21, 22, 48, 49]. The *DARPA* NTIMIT acoustic-phonetic continuous speech corpus contains 6 300 sentences, 10 sentences spoken by each of 630 speakers from eight major dialect regions of the United States. The NTIMIT, or network speech database, is a telephone bandwidth version of the widely-used TIMIT database. Some useful features of NTIMIT (excluding the characteristics that make TIMIT useful):

- Actual telephone channels were used, not simulations or models.
- Telephone channels used for transmission were varied in a "controlled" manner in order to sample various line conditions.
- NTIMIT utterances are time-aligned with TIMIT utterances, allowing the use of existing phonetic, orthographic and other transcriptions.

The NTIMIT speech corpus (and more specifically the TIMIT speech corpus) is one of the standard benchmark corpora for speaker-independent, phoneme- and speech recognition systems.

C.1.1 Format of speech data

The NTIMIT speech corpus is the TIMIT speech corpus transmitted over a telephone network. The TIMIT speech corpus is recorded under good, controlled conditions using a high-quality microphone. The speech is sampled at 16kHz and stored in the standard NIST Sphere format, using 16 bits/sample.

C.1.2 Phoneme set

It was decided to use the well-known, 39-phoneme set proposed by Lee [27]. This set takes the complete set of 64 possible phonetic labels in the NTIMIT corpus and folds it down to 39 main phonemes. All the glottal stops are removed from the transcription labels. The 15 allophones in the complete set are folded into the corresponding phonemes. Certain groups of phonemes are identified where within-group confusions are not counted. These groups are {sil, cl, vcl, epi}, {el, l}, {en, n}, {sh, zh}, {ao, aa}, {ih, ix} and {ah, ax}. This results in 39 separate phonemes. The TIMIT 39-phoneme set is widely used by different researchers to compare different phoneme recognitions systems.

Table C.1: *Reduced NTIMIT-derived phoneme set proposed by Lee* [27].

No.	IPA	ARPABET	Broad phonemic class	Example	Occurrences
1	/æ/	ae	Vowel	bat	3 064
2	/ʌ/	ah	Vowel	but	8 453
3	/ɔ/	ao	Vowel	bought	5 728
4	/a͡ʊ/	aw	Vowel	bout	944
5	/a͡ɪ/	ay	Vowel	bite	2 620
6	/b/	b	Plosive	bee	3 067
7	/t͡ʃ/	ch	Affricate	choke	1 079
8	/d/	d	Plosive	day	3 273
9	/ð/	dh	Fricative	then	3 272
10	/ɾ/	dx	Plosive	muddy	2 498
11	/ɛ/	eh	Vowel	bet	4 524
12		epi	Silence		17 217
13	/ɜ͡r/	er	Vowel	bird	5 830
14	/e͡ɪ/	ey	Vowel	bait	3 073
15	/f/	f	Fricative	fin	3 126
16	/g/	g	Plosive	gay	1 643
17	/h/	hh	Glide	hay	2 221
18	/ɪ/	ih	Vowel	bit	15 557
19	/i/	iy	Vowel	beet	6 436
20	/d͡ʒ/	jh	Affricate	joke	1 308
21	/k/	k	Plosive	key	4 998
22	/l/	l	Glide	lay	7 577
23	/m/	m	Nasal	mom	4 972
24	/n/	n	Nasal	noon	10 176
25	/ŋ/	ng	Nasal	sing	1 598
26	/o/	ow	Vowel	boat	2 253

No.	IPA	ARPABET	Broad phonemic class	Example	Occurrences
27	/ɔɪ/	oy	Vowel	boy	431
28	/o/	p	Plosive	pea	3 545
29	/r/	r	Glide	ray	6 530
30	/s/	s	Fricative	sea	8 348
31	/ʃ/	sh	Fricative	she	1 999
32	/t/	t	Plosive	tea	12 176
33	/θ/	th	Fricative	thin	1 004
34	/ʊ/	uh	Vowel	book	715
35	/u/	uw	Vowel	boot	2 524
36	/v/	v	Fricative	van	2 704
37	/w/	w	Glide	way	3 119
38	/j/	y	Glide	yacht	1 371
39	/z/	z	Fricative	zone	4 918

C.1.3 Data sets

The NTIMIT corpus consists of three types of sentences divided into two non-overlapping groups of training- and testing data. The three types of sentences are the so-called **sa**, **sx** and **si** sentences. The **sa** sentences are the same across all the speakers, the **sx** sentences are read from a list of 450 phonetically-balanced sentences, and the **si** sentences are read from a list of 1 890 phonetically diverse sentences. The **sa** sentences are not used during training or testing because they introduce unfair bias for certain phonemes in certain contexts, which will lead to artificially high recognition scores. This leaves 3 696 sentences from the **sx** and **si** group, which are used for training purposes. The training set consist of 462 different speakers for a total of 116 314 phonemes and 12 680 non-speech events. The rest of the 1 344 sentences from the **sx** and **si** group are used for testing purposes, which consist of 168 different speakers for a total testset of 42 104 phonemes and 4 508 non-speech events. The *a priori* distribution of the train- and testsets are shown in Fig. C.1.

C.2 African Speech Technology speech corpora

The African Speech Technology speech corpora consists of five different South African languages, which include English, Afrikaans, isiXhosa, isiZulu and Sesotho. The speech in each database consists of a mixture of read and spontaneous speech of mother-tongue speakers for a total of 38 to 40 utterances per speaker. The speech is recorded under normal telephone and cellphone conditions.

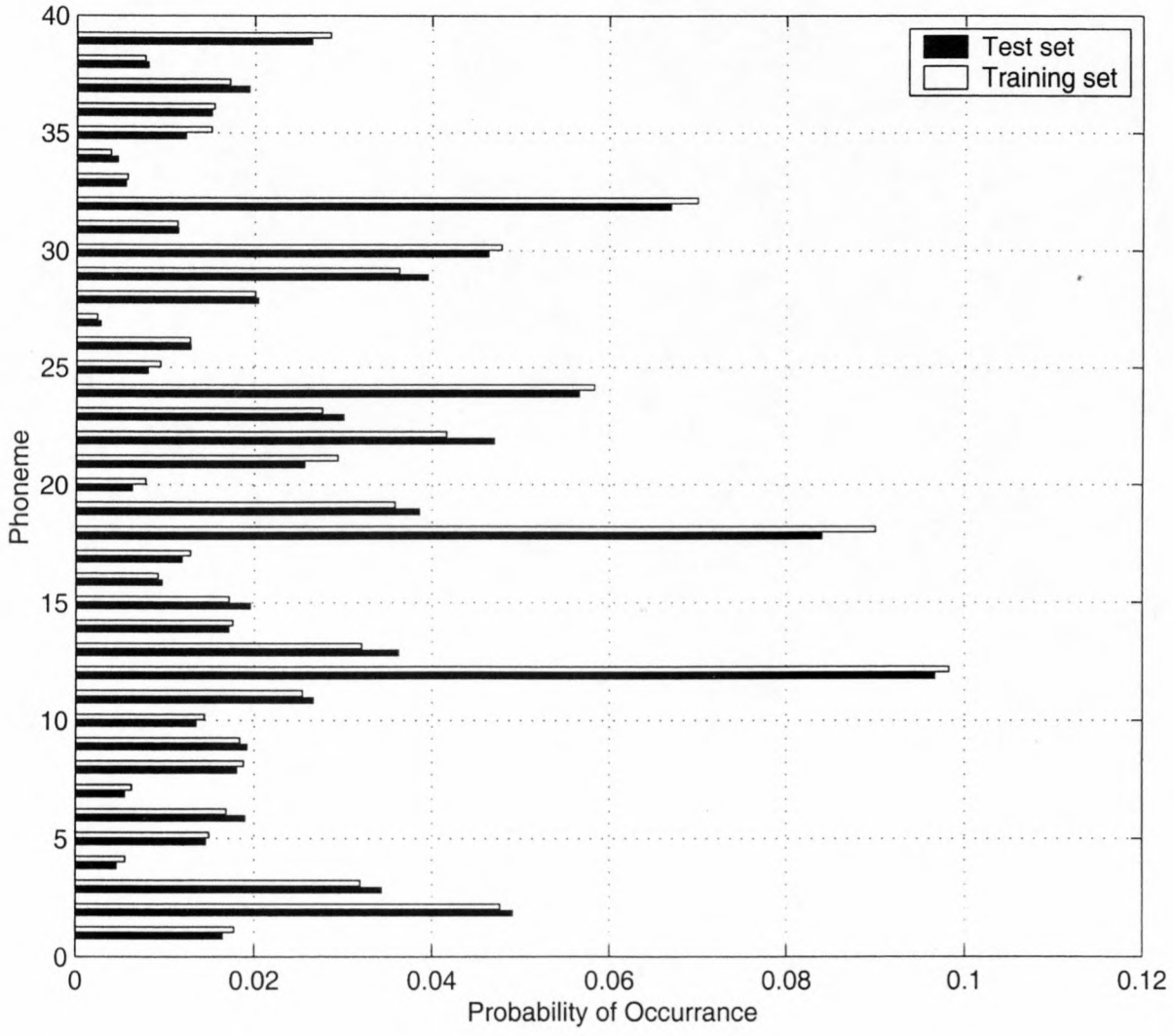

Figure C.1: *A priori distribution of NTIMIT phonemes in training and test sets.*

This thesis focuses on the South African English. The English corpus consists of five different dialects, which include Standard English, Black English, Coloured English, Asian English, Afrikaans English. Each of the different dialects represents a different ethnic group's version of English. For the purpose of this research, only the Standard English database is used, which consists of 287 unique speakers for a total of 11 433 utterances. The specifications of the speech databases are as follows:

- Age: 20-60 years
- Nearly-equal number of male and female speakers
- Call duration: 7-10 minutes
- Nearly-equal mixture of cellphone and telephone calls.

The speech corpora is phonemically and orthographically transcribed. Unlike the NTIMIT speech corpus, the utterances are not time aligned. Therefore, complete (or time-aligned)

transcriptions need to be created using forced alignment of the phonemes and phoneme models initialised from the NTIMIT speech corpus. For more information regarding the contents of the speech corpora, refer to [3, 7].

C.2.1 Format of speech data

The speech data is recorded by using a Dialogic analogue telephony card connected to the South African telephone network. The speech data recorded is stored in raw (headerless) format, containing audio samples at eight bits/sample, and sampled at 8 kHz, mono. Samples are encoded by using mu-law compression.

C.2.2 Phoneme Set

The same phoneme set is used throughout the whole African Speech Technology corpora, but any single database does not contain all of the phonemes.

C.2.2.1 South African English

The Standard English database has only 42 distinct phonemes, excluding the different types of non-speech events such as noise. These 42 distinct phonemes are listed in Table C.2.

Table C.2: *Standard English phoneme set.*

No.	IPA	Broad phonemic class	Example[1]	Occurrences
1	/ɒ/	Vowel	hot	2 740
2	/æ/	Vowel	average	4 384
3	/a͡ʊ/	Diphtong	power	1 121
4	/ɑː/	Vowel	harp	1 765
5	/b/	Plosive	baby	2 752
6	/ɔ͡ɪ/	Diphtong	boy	88
7	/ɔː/	Vowel	bore	2 605
8	/d/	Plosive	death	7 158
9	/ð/	Fricative	this	2 147
10	/ʤ/	Affricate	jug	998
11	/e͡ɪ/	Diphtong	play	4 680
12	/ɛ/	Vowel	nest	6 527
13	/ɜː/	Vowel	turn	1 233
14	/f/	Fricative	four	5 166

[1]These are examples of the phonemes in South African English

No.	IPA	Broad Phonemic Class	Example	Occurrences
15	/ɡ/	Plosive	gun	1 744
16	/h/	Semi-vowel	hand	1 644
17	/ʊ/	Vowel	push	1 273
18	/ɪ/	Vowel	him	9 131
19	/ɪə̯/	Diphtong	here	828
20	/iː/	Vowel	keep	7 486
21	/j/	Semi-vowel	yes	2 764
22	/k/	Plosive	kick	5 425
23	/l/	Semi-vowel	legs	6 228
24	/m/	Nasal	man	4 117
25	/n/	Nasal	not	17 638
26	/ŋ/	Nasal	thing	1 678
27	/p/	Plosive	spit	2 830
28	/ɹ/	Semi-vowel	red	7 039
29	/s/	Fricative	some	10 600
30	/ʃ/	Fricative	shine	1 096
31	/ə/	Vowel	the	15 516
32	/əʊ̯/	Diphtong	hope	4 347
33	/t/	Plosive	total	15 137
34	/θ/	Fricative	thing	3 249
35	/t͡ʃ/	Affricate	chocolate	588
36	/uː/	Vowel	blue	3 669
37	/v/	Fricative	vat	5 134
38	/ɒ/	Vowel	hut	3 848
39	/ɒɪ̯/	Diphtong	fine	4 917
40	/w/	Semi-vowel	west	4 371
41	/z/	Fricative	zero	4 409
42	/ʒ/	Fricative	genre	96
43	[sil]	Silence		18 684
44	[sil]?	Optional Silence		51 758
45	[sta]	Stationary noise		8 912
46	[sta]?	Optional stationary noise		3 990
47	[int]	Intermittent noise		4 957
48	[int]?	Optional intermittent noise		1 974
49	[spk]	Speaker noise		6 713
50	[empty]	Empty		45
51	[ext]			1 387

C.2.3 Data sets

When selecting the training- and testing sets of each database, care is taken to ensure that there are at least 100 occurences of each phoneme in the training set. If this first requirement is met, an attempt is made to include about 30 000 phonemes in the testing set, to make the results on the South African speech corpora comparable to the results on NTIMIT.

C.2.3.1 South African English

The training set consists of 9 275 utterances from 238 different speakers for a total of 159 560 phonemes and 87 292 non-speech events. The testing set consists of 1 813 utterances from different 49 speakers for a total of 30 515 phonemes and 11 072 non-speech events. There is no overlapping of speakers between the training set and the test set. The *a priori* distribution of the training- and test sets are shown in Fig. C.2.

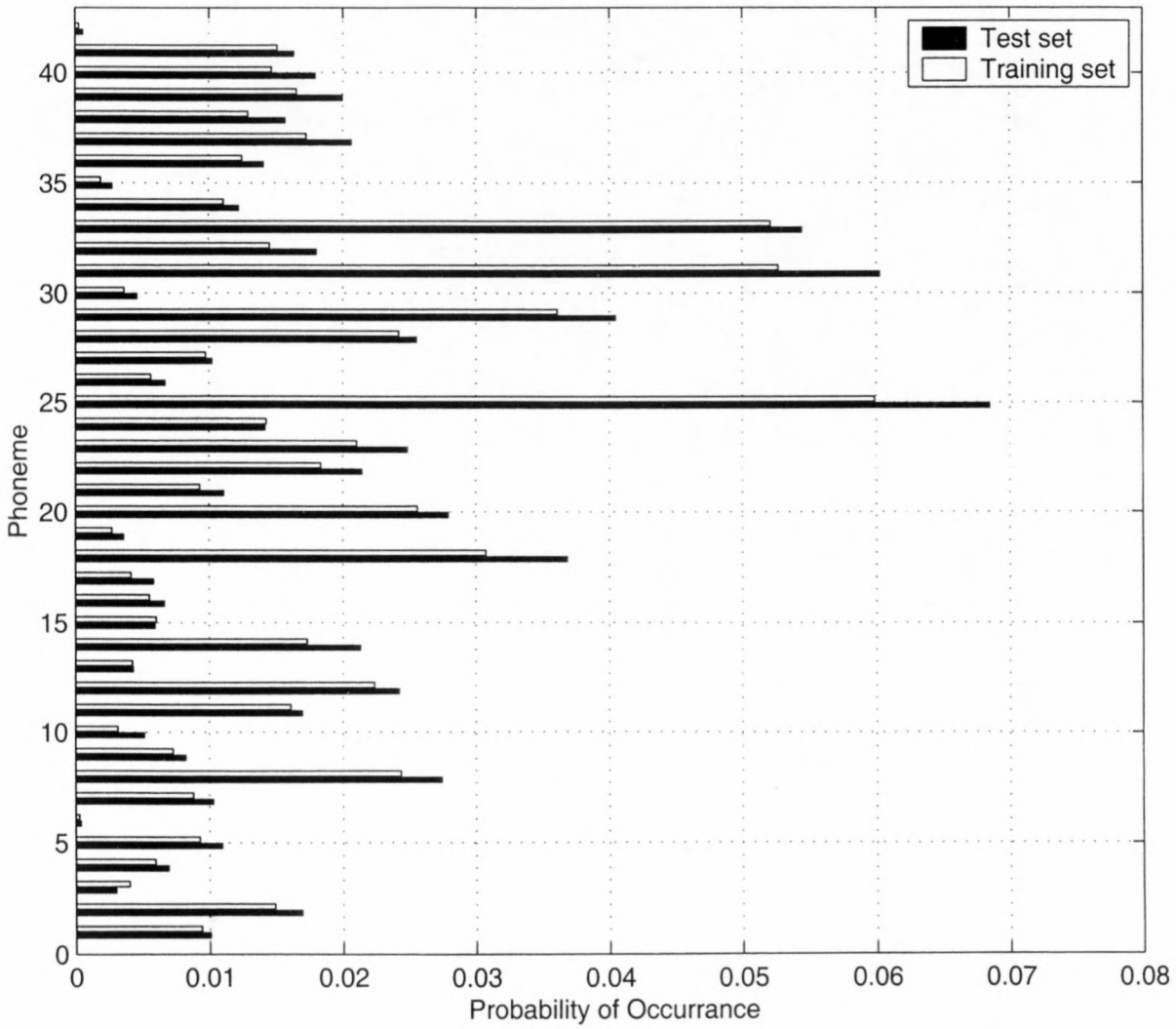

Figure C.2: *A priori distribution of South African English phonemes in the training and test sets.*

Appendix D

Tables of recognition results

D.1 Total number of parameters' calculation

A single diagonal-covariance Gaussian distribution has $2D$ free parameters (where D is the dimension of the pdf.) and a single full-covariance distribution has $D(D+3)/2$ free parameters. A M_{FC} mixture full-covariance GMM has $M_{FC}\left[\frac{D(D+3)}{2}+1\right]$ free parameters, while a M_{DC} mixture diagonal-covariance has $M_{DC}\left[2D+1\right]$ free parameters. Thus, the number of parameters per state distribution is tabulated in Tables D.1 and D.2.

Table D.1: *The total number of parameters of diagonal-covariance Gaussian state distributions.*

Pdf.	Dim.	# P/P[1]
DC-Gaussian	17	34
2 mixture DC GMM	17	70
3 mixture DC GMM	17	105
4 mixture DC GMM	17	140
5 mixture DC GMM	17	175
6 mixture DC GMM	17	210
8 mixture DC GMM	17	280
16 mixture DC GMM	17	560
20 mixture DC GMM	17	665
24 mixture DC GMM	17	840
39 mixture DC GMM	17	1 365
78 mixture DC GMM	17	2 730

[1]Number of parameters per state distribution

Table D.2: *The total number of parameters of full-covariance Gaussian state distributions.*

Pdf.	Dim.	# P/P
FC-Gaussian	17	170
2 mixture FC GMM	17	342
3 mixture FC GMM	17	513
4 mixture FC GMM	17	684
6 mixture FC GMM	17	1 026
8 mixture FC GMM	17	1 368
16 mixture FC GMM	17	2 736
24 mixture FC GMM	17	4 104

The total number of parameters of a spotter or parallel HMM is determined by the summation of the total number of state distribution parameters and the total number of HMM transition parameters.

D.2 Signal processing

Table D.3: *Isolated phoneme recognition results of* 12^{th}*-order MFCCs, augmented with velocity and acceleration features, on the NTIMIT speech corpus.*

Δ	ΔΔ	Pdf.	Dim.	Acc.	Rel. Improv.
		DC-Gauss	12	31.14%	-
√		DC-Gauss	24	37.76%	+21.26%
√	√	DC-Gauss	36	39.44%	+26.65%
		FC-Gauss	12	38.17%	-
√		FC-Gauss	24	48.58%	+27.27%
√	√	FC-Gauss	36	52.48%	+37.49%
		8 mixture DC GMM	12	36.72%	-
√		8 mixture DC GMM	24	47.19%	+28.51%
√	√	8 mixture DC GMM	36	49.98%	+36.11%
		24 mixture DC GMM	12	38.47%	-
√		24 mixture DC GMM	24	51.47%	+33.79%
√	√	24 mixture DC GMM	36	53.83%	+39.93%
		8 mixture FC GMM	12	40.99%	-
√		8 mixture FC GMM	24	57.03%	+39.13%
√	√	8 mixture FC GMM	36	61.34%	+49.65%
		24 mixture FC GMM	12	42.50%	-
√		24 mixture FC GMM	24	58.77%	+38.28%
√	√	24 mixture FC GMM	36	61.93%	+45.72%

D.3 Statistical modelling of HMM state distributions

Table D.4: *Isolated phoneme recognition results on NTIMIT for different state output probability distributions. There is a total of 312 HMM transition parameters.*

Features	Pdf.	Dim.	Acc.	Params.[2]
12^{th}-order MFCC	DC-Gaussian	17	43.67%	4 290
12^{th}-order MFCC	4 mixture DC GMM	17	50.21%	16 692
12^{th}-order MFCC	8 mixture DC GMM	17	52.60%	33 072
12^{th}-order MFCC	16 mixture DC GMM	17	54.16%	65 832
12^{th}-order MFCC	24 mixture DC GMM	17	55.22%	98 592
12^{th}-order MFCC	FC-Gaussian	17	50.36%	20 202
12^{th}-order MFCC	4 mixture FC GMM	17	55.69%	80 340
12^{th}-order MFCC	8 mixture FC GMM	17	57.86%	160 368
12^{th}-order MFCC	16 mixture FC GMM	17	59.31%	320 424
12^{th}-order MFCC	24 mixture FC GMM	17	60.35%	480 480

Table D.5: *Isolated phoneme recognition results on NTIMIT comparing diagonal- and full-covariance state pdfs with the nearly equal number of parameters. There is a total of 312 HMM transition parameters.*

Features	Pdf.	# P/P	Acc.	Params.
12^{th}-order MFCC	FC-Gaussian	170	50.36%	20 202
12^{th}-order MFCC	5 mixture DC GMM	175	51.01%	20 787
12^{th}-order MFCC	4 mixture FC GMM	684	55.69%	80 340
12^{th}-order MFCC	20 mixture DC GMM	665	54.72%	78 117
12^{th}-order MFCC	8 mixture FC GMM	1 368	57.88%	160 368
12^{th}-order MFCC	39 mixture DC GMM	1 365	56.23%	160 017
12^{th}-order MFCC	16 mixture FC GMM	2 736	59.31%	320 424
12^{th}-order MFCC	78 mixture DC GMM	2 730	57.82%	319 722

[2]Total number of parameters

D.4 Context-dependent modelling

Table D.6: *Isolated phoneme recognition results on NTIMIT using left- and right-context biphones to model context dependency. The total of HMM transition parameters for monophone, LC biphone and RC biphone are respectively 312, 12 620 and 12 620.*

Features	Modelling	Pdf.	Acc.	Params.
12^{th}-order MFCC	Monophones	2 mixture DC GMM	44.48%	18 286
12^{th}-order MFCC	Monophones	3 mixture DC GMM	49.16%	12 597
12^{th}-order MFCC	Monophones	4 mixture DC GMM	50.21%	16 692
12^{th}-order MFCC	Monophones	6 mixture DC GMM	51.41%	24 882
12^{th}-order MFCC	Monophones	8 mixture DC GMM	52.61%	33 072
12^{th}-order MFCC	Monophones	24 mixture DC GMM	55.22%	98 592
12^{th}-order MFCC	Monophones	FC-Gaussian	50.35%	20 202
12^{th}-order MFCC	Monophones	2 mixture FC GMM	53.73%	40 326
12^{th}-order MFCC	Monophones	3 mixture FC GMM	54.65%	60 333
12^{th}-order MFCC	Monophones	4 mixture FC GMM	55.69%	80 340
12^{th}-order MFCC	Monophones	6 mixture FC GMM	57.06%	120 354
12^{th}-order MFCC	Monophones	8 mixture FC GMM	57.86%	160 386
12^{th}-order MFCC	Monophones	24 mixture FC GMM	60.35%	480 480
12^{th}-order MFCC	LC Biphones	2 mixture DC GMM	53.62%	99 560
12^{th}-order MFCC	LC Biphones	3 mixture DC GMM	54.57%	143 030
12^{th}-order MFCC	LC Biphones	4 mixture DC GMM	55.53%	186 500
12^{th}-order MFCC	LC Biphones	6 mixture DC GMM	56.79%	273 440
12^{th}-order MFCC	LC Biphones	FC-Gaussian	56.78%	223 760
12^{th}-order MFCC	LC Biphones	2 mixture FC GMM	58.77%	437 384
12^{th}-order MFCC	LC Biphones	3 mixture FC GMM	59.13%	649 766
12^{th}-order MFCC	LC Biphones	4 mixture FC GMM	59.41%	862 148
12^{th}-order MFCC	LC Biphones	6 mixture FC GMM	59.54%	1 286 912
12^{th}-order MFCC	RC Biphones	2 mixture DC GMM	53.65%	99 700
12^{th}-order MFCC	RC Biphones	3 mixture DC GMM	54.88%	143 240
12^{th}-order MFCC	RC Biphones	4 mixture DC GMM	55.63%	186 780
12^{th}-order MFCC	RC Biphones	6 mixture DC GMM	56.54%	273 860
12^{th}-order MFCC	RC Biphones	FC-Gaussian	56.29%	224 100
12^{th}-order MFCC	RC Biphones	2 mixture FC GMM	58.40%	438 068
12^{th}-order MFCC	RC Biphones	3 mixture FC GMM	58.96%	650 792
12^{th}-order MFCC	RC Biphones	4 mixture FC GMM	59.44%	863 516
12^{th}-order MFCC	RC Biphones	6 mixture FC GMM	59.72%	1 288 964

Table D.7: *Isolated phoneme recognition results on NTIMIT, using triphones to model context dependency. The total of HMM transition parameters for triphones are 153 080.*

Features	Pdf.	$\tau = 350$		$\tau = 700$	
		Acc.[3]	Params.	Acc.	Params.
12^{th}-order MFCC	2 mixture DC GMM	56.52%	293 640	55.92%	225 040
12^{th}-order MFCC	3 mixture DC GMM	57.16%	363 920	57.23%	261 020
12^{th}-order MFCC	4 mixture DC GMM	57.67%	434 200	57.66%	297 000
12^{th}-order MFCC	6 mixture DC GMM	58.35%	574 760	58.21%	368 960
12^{th}-order MFCC	FC-Gaussian	59.20%	494 440	59.28%	327 840
12^{th}-order MFCC	2 mixture FC GMM	60.12%	839 816	60.65%	504 656
12^{th}-order MFCC	3 mixture FC GMM	59.90%	1 183 184	60.72%	680 444
12^{th}-order MFCC	4 mixture FC GMM	59.60%	1 526 552	60.67%	856 232
12^{th}-order MFCC	6 mixture FC GMM	58.99%	2 213 288	60.57%	1 207 808

D.5 Continuous phoneme recognition: NTIMIT

Table D.8: *Continuous phoneme recognition results on NTIMIT, using monophones to model context dependency.* 12^{th}*-order MFCCs are used as features. There is a total of 393 HMM transition parameters.*

Pdf.	Subs.	Ins.	Del.	Cor.	Acc.	Params.
2 mixture DC GMM	47.08%	3.12%	12.45%	40.47%	37.65%	8 583
3 mixture DC GMM	43.29%	2.87%	12.80%	43.91%	41.33%	12 678
4 mixture DC GMM	42.47%	2.84%	12.64%	44.90%	42.34%	16 773
6 mixture DC GMM	41.33%	2.91%	12.45%	46.22%	43.59%	24 963
FC-Gaussian	41.27%	2.46%	13.85%	44.87%	42.71%	20 283
2 mixture FC GMM	40.14%	3.09%	11.71%	48.14%	45.33%	40 407
3 mixture FC GMM	38.96%	3.11%	11.54%	49.50%	46.66%	60 414
4 mixture FC GMM	38.29%	3.11%	11.31%	50.39%	47.54%	80 421
6 mixture FC GMM	37.39%	3.04%	11.14%	51.48%	48.69%	120 435
24 mixture FC GMM	34.93%	3.39%	10.61%	54.46%	51.32%	480 561

[3] τ – The minimum node state occupancy used when constructing the decision trees.

Table D.9: *Continuous phoneme recognition results on NTIMIT, using monophones and left-context biphones to model context dependency.* 12^{th}*-order MFCCs are used as features. There is a total of 104 830 HMM transition parameters.*

Pdf.	Subs.	Ins.	Del.	Cor.	Acc.	Params.
2 mixture DC GMM	42.87%	5.15%	9.00%	48.14%	43.19%	199 960
3 mixture DC GMM	41.13%	4.93%	9.23%	49.64%	44.94%	247 525
4 mixture DC GMM	40.20%	4.91%	9.11%	50.69%	46.00%	295 090
6 mixture DC GMM	39.40%	5.14%	8.61%	51.99%	47.04%	390 220
FC-Gaussian	38.82%	4.82%	8.60%	52.58%	47.95%	335 860
2 mixture FC GMM	37.31%	4.94%	8.32%	54.38%	49.61%	569 608
3 mixture FC GMM	36.31%	4.96%	8.29%	55.40%	50.62%	801 997
4 mixture FC GMM	35.67%	4.45%	8.80%	55.53%	51.28%	1 034 386
6 mixture FC GMM	35.07%	4.23%	9.23%	55.70%	51.69%	1 499 164

Table D.10: *Continuous phoneme recognition results on NTIMIT, using monophones and right-context biphones to model context dependency.* 12^{th}*-order MFCCs are used as features. There is a total of 104 828 HMM transition parameters.*

Pdf.	Subs.	Ins.	Del.	Cor.	Acc.	Params.
2 mixture DC GMM	40.73%	4.06%	10.29%	48.98%	45.18%	200 098
3 mixture DC GMM	39.40%	4.08%	10.08%	50.51%	46.69%	247 733
4 mixture DC GMM	38.77%	4.09%	10.00%	51.23%	47.39%	295 368
6 mixture DC GMM	38.14%	4.42%	9.36%	52.50%	48.31%	390 538
FC-Gaussian	38.02%	3.96%	9.71%	52.27%	48.55%	336 198
2 mixture FC GMM	36.25%	4.30%	9.03%	54.72%	50.63%	570 290
3 mixture FC GMM	35.48%	4.31%	8.97%	55.55%	51.45%	803 021
4 mixture FC GMM	34.79%	4.22%	9.30%	55.91%	51.91%	1 035 752
6 mixture FC GMM	34.41%	3.88%	9.32%	56.27%	52.61%	1 501 214

Table D.11: *Continuous recognition accuracies on NTIMIT, using monophones and biphones to model context dependency and* 12^{th}*-order MFCCs as feature vectors. There is a total of 1 801 653 HMM transition parameters.*

Pdf.	Subs.	Ins.	Del.	Cor.	Acc.	Params.
2 mixture DC GMM	40.02%	4.49%	9.70%	50.28%	46.04%	1 983 863
3 mixture DC GMM	38.93%	4.53%	9.29%	51.79%	47.48%	2 074 968
4 mixture DC GMM	38.40%	4.50%	9.03%	52.57%	48.29%	2 166 073
6 mixture DC GMM	37.51%	5.02%	8.48%	54.01%	49.17%	2 348 283
FC-Gaussian	37.21%	4.98%	8.30%	54.49%	49.69%	2 244 163
2 mixture FC GMM	35.50%	5.07%	7.88%	56.62%	51.71%	2 691 879
3 mixture FC GMM	34.90%	5.20%	7.61%	57.50%	52.43%	3 136 992
4 mixture FC GMM	34.51%	4.96%	7.80%	57.70%	52.88%	3 582 105
6 mixture FC GMM	34.24%	4.67%	7.99%	57.77%	53.26%	4 472 331

Table D.12: *Continuous recognition accuracies on NTIMIT, using mono-, bi-, and triphones to model context dependency and* 12^{th}*-order MFCCs as feature vectors. There is a total of 3 286 206 HMM transition parameters.*

Pdf.	Subs.	Ins.	Del.	Cor.	Acc.	Params.
2 mixture DC GMM	37.77%	4.01%	10.28%	51.95%	48.20%	3 608 976
3 mixture DC GMM	36.96%	4.05%	9.86%	53.18%	49.38%	3 770 361
4 mixture DC GMM	36.14%	4.07%	9.66%	54.20%	50.37%	3 931 746
6 mixture DC GMM	35.58%	4.39%	8.99%	55.43%	51.25%	4 254 516
FC-Gaussian	34.66%	4.26%	8.96%	56.38%	52.33%	4 070 076
2 mixture FC GMM	33.77%	4.43%	8.35%	57.88%	53.63%	4 863 1683
3 mixture FC GMM	33.17%	4.41%	8.43%	58.40%	54.18%	5 651 649
4 mixture FC GMM	32.77%	4.17%	8.78%	58.44%	54.47%	6 440 130
6 mixture FC GMM	32.77%	3.92%	9.07%	58.16%	54.45%	8 017 092

D.6 Continuous phoneme recognition: S.A. English

Table D.13: *Continuous recognition accuracies on South African English, using monophones to model context dependency and* 12^{th}*-order MFCCs as feature vectors. There is a total of 1 161 HMM transition parameters.*

Pdf.	Subs.	Ins.	Del.	Cor.	Acc.	Params.
2 mixture DC GMM	41.61%	6.35%	10.73%	47.66%	39.89%	22 293
3 mixture DC GMM	39.05%	7.07%	10.33%	50.62%	41.89%	26 631
4 mixture DC GMM	37.92%	7.18%	10.44%	51.64%	42.77%	31 041
6 mixture DC GMM	36.78%	7.20%	10.38%	52.83%	43.94%	39 861
FC-Gaussian	35.17%	6.94%	11.60%	53.24%	45.00%	33 461
2 mixture FC GMM	33.26%	5.57%	10.99%	55.75%	46.83%	56 493
3 mixture FC GMM	32.73%	7.42%	10.80%	56.47%	47.39%	78 039
4 mixture FC GMM	31.99%	7.45%	10.81%	57.20%	48.06%	99 585
6 mixture FC GMM	31.11%	7.40%	10.95%	57.94%	48.91%	142 677

Table D.14: *Continuous recognition accuracies on South African English, using monophones and left-context biphones to model context dependency and* 12^{th}*-order MFCCs as feature vectors. There is a total of 98 040 HMM transition parameters.*

Pdf.	Subs.	Ins.	Del.	Cor.	Acc.	Params.
2 mixture DC GMM	32.24%	9.24%	8.80%	58.96%	48.00%	152 840
3 mixture DC GMM	31.19%	9.04%	8.58%	60.23%	49.52%	174 120
4 mixture DC GMM	30.21%	9.30%	8.65%	61.14%	50.09%	195 400
6 mixture DC GMM	29.65%	9.21%	8.77%	61.59%	50.69%	237 960
FC-Gaussian	27.94%	9.14%	8.80%	63.26%	52.49%	213 640
2 mixture FC GMM	26.93%	9.22%	8.89%	64.18%	53.31%	318 216
3 mixture FC GMM	26.65%	9.26%	8.92%	64.42%	53.52%	422 184
4 mixture FC GMM	25.88%	9.29%	9.06%	65.06%	54.15%	526 152
6 mixture FC GMM	25.60%	8.75%	9.26%	65.14%	54.93%	734 088

Table D.15: *Continuous recognition accuracies on South African English, using monophones and right-context biphones to model context dependency and* 12^{th}*-order MFCCs as feature vectors. There is a total of 100 486 HMM transition parameters.*

Pdf.	Subs.	Ins.	Del.	Cor.	Acc.	Params.
2 mixture DC GMM	31.59%	8.43%	9.77%	58.64%	48.83%	152 136
3 mixture DC GMM	31.06%	8.79%	9.53%	59.41%	49.13%	171 841
4 mixture DC GMM	29.60%	8.62%	9.73%	60.67%	50.64%	191 546
6 mixture DC GMM	28.69%	8.60%	9.69%	61.62%	51.63%	230 956
FC-Gaussian	27.03%	7.80%	9.93%	63.04%	54.15%	208 436
2 mixture FC GMM	26.31%	8.78%	9.56%	64.13%	53.93%	305 272
3 mixture FC GMM	26.30%	8.64%	9.52%	64.18%	54.18%	401 545
4 mixture FC GMM	25.59%	8.62%	9.68%	64.74%	54.76%	497 818
6 mixture FC GMM	25.36%	8.37%	9.68%	64.97%	55.34%	690 364


engelbrecht_automatic_2004